PCT/KR 2004/001839
RO/KR 23.07.2004

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2003-0051714
Application Number

출 원 년 월 일 : 2003년 07월 26일
Date of Application JUL 26, 2003

출 원 인 : (주)이진공작 외 3명
Applicant(s) BINACRAFT, et al.

2004 년 07 월 23 일

특 허 청
COMMISSIONER

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【제출일자】	2003.07.26
【발명의 명칭】	클라이언트 시스템의 동작 동기화 장치와 방법 및 이를 이용한 통신망을 통한 상호 작용 시스템 및 게임방법
【발명의 영문명칭】	Apparatus and method for syncronizing motion of client system and interaction system and game method using it on network
【출원인】	
【명칭】	(주)이진공작
【출원인코드】	1-2003-028395-9
【출원인】	
【성명】	최강인
【출원인코드】	4-2003-028394-7
【출원인】	
【성명】	안성수
【출원인코드】	4-2003-028393-1
【출원인】	
【성명】	조국영
【출원인코드】	4-1999-042778-1
【대리인】	
【명칭】	특허법인 우린
【대리인코드】	9-2003-100041-1
【지정된변리사】	이병길
【포괄위임등록번호】	2003-051500-3
【포괄위임등록번호】	2003-051504-2
【포괄위임등록번호】	2003-051503-5
【포괄위임등록번호】	2003-051889-0
【발명자】	
【성명】	최강인
【출원인코드】	4-2003-028394-7

102[redacted]51714

출력 일자: 2004/7/30

【발명자】

【성명】 안성수

【출원인코드】 4-2003-028393-1

【발명자】

【성명】 조국영

【출원인코드】 4-1999-042778-1

【발명자】

【성명의 국문표기】 장완호

【성명의 영문표기】 JANG,Wan-ho

【주민등록번호】 670706-1000810

【우편번호】 431-050

【주소】 경기도 안양시 동안구 비산동 1101-8 샛별아파트 305동 209호

【국적】 KR

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
특허법인 우린 (인)

【수수료】

【기본출원료】	20 면	29,000 원
【가산출원료】	70 면	70,000 원
【우선권주장료】	0 건	0 원
【심사청구료】	46 항	1,581,000 원
【합계】	1,680,000 원	
【감면사유】	소기업 (70%감면)	
【감면후 수수료】	504,000 원	

【첨부서류】 1. 요약서·명세서(도면)_1통 2.소기업임을 증명하는 서류_1통

【요약서】

【요약】

이 발명은 임의의 클라이언트 시스템에서 키입력부를 통해 입력되는 이벤트에 의해 동작하는 구조체의 동작을 단위시간에 동기시켜 표시하는 동작 동기화 장치 및 방법에 관한 것이며, 아울러 상기한 동작 동기화 장치 및 방법을 이용하여 통신망을 통해 임의의 두 클라이언트 시스템 간에 동작의 상호작용이 가능하도록 하는 시스템 및 게임방법에 관한 것이다.

이 발명에 따른 동작 동기화 방법은, 각종 온라인 및 웹상에서 다수의 클라이언트 시스템이 접속되어 구현되는 가상공간 상에서, 상기 다수의 클라이언트 시스템들을 표준 시간으로 동기화하고, 상기 다수의 클라이언트 시스템들 각각에서 발생한 이벤트들을 상대방 클라이언트들에게 전송하고, 상기 일정한 기준구간 동안에 발생한 이벤트들을 상기 표준시간에 동기시켜 화면에 표시한다.

이 발명에 따르면 3차원으로 구현되는 컴퓨터 그래픽스의 영상에서 구조체의 자유로운 동작 표현뿐만 아니라 복수의 링크를 갖는 구조체 간의 상호간섭을 고려한 구조체의 동작을 보다 단순하고 평이한 조작으로 구현할 수 있다. 또한, 3차원 그래픽스 영상을 통하여 제공되는 구조체 동작에서 실제와 유사한 물리적 특성을 구현할 수 있으며, 구조체 전체가 자연스러운 자세를 유지하면서도 연속 동작이 가능하게 된다.

【대표도】

도 2

【색인어】

이벤트, 시간설정, 동기화, 단위동작, 패킷, 동시성, 구조체, 가상공간

【명세서】

【발명의 명칭】

클라이언트 시스템의 동작 동기화 장치와 방법 및 이를 이용한 통신망을 통한 상호 작용 시스템 및 게임방법{Apparatus and method for syncronizing motion of client system and interaction system and game method using it on network}

【도면의 간단한 설명】

도 1은 이 발명의 한 실시 예에 따른 동작 동기화 장치를 포함한 클라이언트 시스템의 개략적인 구성도,

도 2는 이 발명의 한 실시 예에 따른 동작 동기화 장치의 기능 블록도,

도 3은 이 발명의 한 실시 예에 따른 동작 동기화 장치를 이용한 통신망을 통한 상호작용 시스템을 도시한 기능 블록도,

도 4는 클라이언트 시스템에서의 댄스 게임방법을 도시한 동작 흐름도,

도 5는 리더인 제1클라이언트 시스템에서의 동작 입력 처리 서브루틴을 도시한 동작 흐름도,

도 6은 팔로우어인 제2클라이언트 시스템에서의 동작 입력 처리 서브루틴을 도시한 동작 흐름도,

도 7은 사용자의 모니터에 표시되는 GUI화면의 일 예시도,

도 8은 댄스게임이 진행되는 동안에 리더인 제1클라이언트 시스템에 디스플레이되는 예시 화면,

도 9는 댄스게임이 진행되는 동안에 팔로우어인 제2클라이언트 시스템에 디스플레이되는 예시 화면이다.

<도면의 주요 부분에 대한 부호의 설명>

100 : 입력부	101 : 입력기
110 : 연산처리부	111 : 이벤트처리기
112 : 액세스데이터처리기	113 : GUI처리기
120 : 동기화부	121 : DB처리기
122 : 가상공간처리기	123 : 개인정보처리기
130 : 인터페이스부	131 : 동작진행처리기
140 : 출력부	141 : 출력기

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<18> 이 발명은 임의의 클라이언트 시스템에서 키입력부를 통해 입력되는 이벤트에 의해 동작하는 구조체의 동작을 단위시간에 동기시켜 표시하는 동작 동기화 장치 및 방법에 관한 것이다. 또한, 이 발명은 상기한 클라이언트 시스템의 동작 동기화 장치 및 방법을 이용하여 통신망을 통해 임의의 두 클라이언트 시스템간에 동작의 상호작용이 가능하도록 하는 시스템 및 게임방법에 관한 것이기도 하다.

<19> 최근 컴퓨터 그래픽스를 사용하여 애니메이션, 3차원 동영상 등과 같은 영상표현 방법이 발달하고 있다. 컴퓨터 그래픽스에서는 키 프레임에 해당되는 데이터를 작성하여 사용하는데, 키 프레임간의 데이터는 스플라인 보간이나 폴리곤 보간 등에 의해 자동적으로 생성한다.

<20> 그러나, 이러한 컴퓨터 그래픽스 데이터 처리방법은 단일 구조체에 적용할 수 있을 뿐이고, 타 구조체와의 상호간섭작용이 있는 경우에는 적용할 수 없다. 여기서, 구조체라 함은 인간이나 동물과 같이 관절 및 관절의 회전범위, 평형유지 동작 등에 의하여 동작에 있어서 균형을 유지하는 물체를 말한다.

<21> 컴퓨터 그래픽스에서, 구조체의 동작을 사실적으로 표현하기 위해서는 실제 동작에서의 중요한 부분을 파악하고 이를 구조체에 대입해야 한다. 따라서, 각각의 동작에 대해 해당 동작의 특성을 분석하고 중요도를 지정해야 한다. 이때 두 구조체간 상호간섭 작용이 있는 경우에는, 하나의 구조체가 타 구조체의 미래의 동작을 모르는 상태에서 임의의 동작에 대한 중요도를 파악해야 하므로, 중요도가 올바르게 지정되지 않아서 구조체의 동작이 어색하게 나타나는 문제점이 발생한다.

<22> 이러한 문제점을 해결하기 위하여, 결과적인 구조체의 동작에서 포착된 동작의 중요한 부분을 보존하는 방법을 이용한 역 운동학적(inverse kinematics) 모델 등이 제안되었다.

<23> 그러나, 이러한 방법 역시 각 구조체간의 상호간섭을 고려하지 않은 단독 구조체에 국한되어 있으며 상호간섭을 고려할 경우는 각 구조체간의 간섭시 동작에 대한 예측문제를 해결할 수가 없기 때문에 다수의 구조체간에는 적용하지 못한다.

<24> 즉, 현재까지 공개된 기술만으로는 다수의 클라이언트간에 구현되는 2개 이상의 구조체 간의 상호간섭을 고려한 동기화를 전혀 구현할 수가 없었다.

【발명이 이루고자 하는 기술적 과제】

<25> 이로부터, 상기하는 종래기술의 문제점을 해결하기 위하여 안출된 이 발명의 목적은, 키 입력부를 통해 입력되는 이벤트에 의해 움직이는 구조체의 동작을 단위시간에 동기시켜 표시하는 동작 동기화 장치 및 방법을 제공하기 위한 것이다.

<26> 또한, 이 발명의 다른 목적은 상기한 클라이언트 시스템의 동작 동기화 장치 및 방법을 이용하여, 통신망으로 연결된 임의의 두 클라이언트 시스템에 종속된 구조체의 동작의 상호작용이 가능하도록 하는 시스템을 제공하기 위한 것이다.

【발명의 구성】

<27> 상기한 목적을 달성하기 위한 이 발명에 따른 클라이언트 시스템의 동작 동기화 장치는, 사용자의 요구에 따라 이벤트를 발생시키는 입력부와, 상기 입력부에서 발생한 이벤트를 표준시간과 비교하여 시간차를 보정하도록 하는 동기화부와, 상기 입력부에서 발생한 이벤트가 상기 표준시간에 동기화되도록 연산 및 시스템 제어하는 연산처리부와, 상기 표준시간에 동기된 이벤트에 따른 영상과 음원이 출력되도록 정합하는 인터페이스부와, 상기 인터페이스부를 통해 출력되는 영상과 음원을 출력하는 출력부를 구비한 것을 특징으로 한다.

<28> 바람직하게는, 상기 입력부는 키보드, 마우스, 트랙 볼, 조이스틱, 터치스크린 및 핸드폰 키패드 중 하나 또는 다수의 조합인 것이어도 좋다.

<29> 바람직하게는, 상기 입력부는 카메라나 센서를 통한 직접행위입력장치 또는 마이크와 같은 음성입력장치인 것이어도 좋다.

<30> 바람직하게는, 상기 동기화부는 상기 표준시간을 설정하는 시간설정부와, 상기 시간설정부에서 설정된 표준시간과 이벤트 전송시간을 계산하여 정보 전송에 따른 이벤트의 시간차를 제거하는 시간동기부와, 구조체를 구성하는 관절 및 이에 의한 이벤트의 동작에 의한 단위동작을 구분하여 진행, 회전, 평행유지 등의 동작을 진행할 수 있도록 하는 단위동작설정부와, 사용자간에 원격으로 접속되어 있는 하나의 구조체와 다른 구조체와의 상호간섭을 고려하기 위해, 설정된 표준시간을 바탕으로 각 사용자에게 동기화된 시간을 토대로 구조체간의 이벤트 및 동작발생의 동시성을 구현하는 단위동작동기부를 구비한 것이어도 좋다.

<31> 바람직하게는, 상기 시간설정부는 월드타임코드(WTC)를 상기 표준시간으로 설정하는 것이어도 좋다.

<32> 바람직하게는, 상기 단위동작설정부에서의 데이터 생성방식은 각각의 동작에 의한 장면에 해당되는 프레임 정점위치와 데이터를 저장하고 있다가 보간법에 의하여 계산하는 것이어도 좋다.

<33> 바람직하게는, 상기 단위동작설정부에서의 데이터 생성방식은 구조체를 여러 개로 나누어서 각각의 관계를 규정하고, 나누어진 구조체의 데이터를 매 프레임 또는 변화되는 프레임마다 지정하여 사용하는 것이어도 좋다.

<34> 바람직하게는, 상기 단위동작설정부에서의 데이터 생성방식은 골격이라는 관절단위의 구조체 데이터를 토대로 각각의 관계를 규정한 계층적 구조를 가지고 위치값에 따라 움직이는 것이어도 좋다.

<35> 바람직하게는, 상기 단위동작동기부는 단위동작을 맞추기 위해서 부수적으로 음원을 사용하는 것이어도 좋다.

<36> 바람직하게는, 상기 음원은 WAV, MP3, WMA, MIDI 중 하나인 것이어도 좋다.

<37> 바람직하게는, 상기 단위동작동기부는 음원의 진행시간에 맞추어 표준시간을 설정하고 상기 표준시간을 단위동작과 동기시키는 것이어도 좋다.

<38> 바람직하게는, 상기 출력부는 화상출력기와 음원출력기를 구비한 것이어도 좋다.

<39> 바람직하게는, 상기 화상출력기는 모니터, HUD(Head Up Display)장치, LCD 패널인 것이어도 좋다.

<40> 바람직하게는, 상기 음원출력기는 스피커인 것이어도 좋다.

<41> 바람직하게는, 상기 화상출력기는 입체로 구성된 대상물과의 송수신하여 상기 입체 대상물을 통해 입출력 매개상태를 확인하는 것이어도 좋다.

<42> 또한, 이 발명에 따른 동작 동기화 방법은, 각종 온라인 및 웹상에서 다수의 클라이언트 시스템이 접속되어 구현되는 가상공간 상에서, 상기 다수의 클라이언트 시스템들을 표준시간으로 동기화하고, 상기 다수의 클라이언트 시스템들 각각에서 발생한 이벤트들을 상대방 클라이언트들에게 전송하고, 상기 일정한 기준구간 동안에 발생한 이벤트들을 상기 표준시간에 동기시켜 화면에 표시하는 것을 특징으로 한다.

<43> 바람직하게는, 상기 가상공간의 시간동기코드를 상기 각 클라이언트 시스템에 동기화한 후 상기 클라이언트 시스템으로부터 입력된 단위동작을 단위동작입력시간 종료후에 시간설정코드에 맞추어 동시에 실행하는 제 16 항에 있어서, 상기 가상공간의 시간동기코드를 상기 각 클

라이언트 시스템에 동기화한 후 상기 클라이언트 시스템으로부터 입력된 단위동작을 단위동작 입력시간 종료후에 시간설정코드에 맞추어 동시에 실행하는 것이어도 좋다.

<44> 바람직하게는, 상기 시간동기코드는 월드타임코드인 것이어도 좋다.

<45> 바람직하게는, 상기 구조체는 상기 클라이언트 시스템으로부터 입력되는 2이상의 이벤트에 따른 상호간섭에 의하여 하나의 형태로 진행되는 것이어도 좋다.

<46> 바람직하게는, 상기 동기화방법은 하나의 서버시스템과 다수의 클라이언트 시스템에 의한 서버/클라이언트 방식으로 운영되는 것이어도 좋다.

<47> 바람직하게는, 상기 동기화방법은 다수의 클라이언트 시스템에 의한 피어투피어방식으로 운영되는 것이어도 좋다.

<48> 바람직하게는, 상기 피어투피어 방식은 정보공유형과 자원공유형 중 하나 또는 둘의 결합을 통해 서비스되는 것이어도 좋다.

<49> 바람직하게는, 상기 피어투피어 방식은 핑(Ping), 퐁(Pong), 쿼리(Query), 쿼리히트(Queryhit), 푸쉬(Push) 등과 같은 스트립터 중 하나 또는 다수를 사용하는 것이어도 좋다.

<50> 바람직하게는, 상기 클라이언트 시스템은 별도의 메모리를 가지고 온라인 혹은 2인 이상 게임이 가능한 PS2 나 X-box, GameCube와 같은 비디오 게임기를 포함하는 것이어도 좋다.

<51> 또한, 이 발명에 따른 상호작용 시스템은, 각종 온라인 및 웹상에서 표준시간으로 동기화되고, 일정한 기준구간 동안에 발생한 이벤트들을 상호간에 전달하며 상기 표준시간에 동기시켜 구조체를 움직이는 다수의 클라이언트 시스템을 포함한 것을 특징으로 한다.

<52> 바람직하게는, 상기 클라이언트 시스템은 사용자의 요구에 따라 이벤트를 발생시키는 입력부와, 상기 입력부에서 발생한 이벤트를 표준시간과 비교하여 시간차를 보정하도록 하는 동

기화부와, 상기 입력부에서 발생한 이벤트가 상기 표준시간에 동기화되도록 연산 및 시스템 제어하는 연산처리부와, 상기 표준시간에 동기된 이벤트에 따른 영상과 음원이 출력되도록 정합하는 인터페이스부와, 상기 인터페이스부를 통해 출력되는 영상과 음원을 출력하는 출력부를 구비한 것이어도 좋다.

<53> 바람직하게는, 상기 동기화부는 DB처리기와, 가상공간처리기와 개인정보처리기를 구비한 것이어도 좋다.

<54> 바람직하게는, 상기 연산처리부는 이벤트처리기와 액세스데이터처리기와 GUI처리기를 구비한 것이어도 좋다.

<55> 바람직하게는, 상기 인터페이스부는 동작진행처리기를 구비한 것이어도 좋다.

<56> 또한, 이 발명에 따른 게임방법은, 각종 온라인 및 웹상에서 다수의 클라이언트 시스템이 접속되어 구현되는 가상공간 상에서, 상기 다수의 클라이언트 시스템들을 표준시간으로 동기화하고, 상기 다수의 클라이언트 시스템들 각각에서 발생한 이벤트들을 상대방 클라이언트들에게 전송하고, 상기 일정한 기준구간 동안에 발생한 이벤트들에 따른 단위동작을 상기 표준시간에 동기시켜 화면에 표시하여 댄스 게임을 서비스한다.

<57> 바람직하게는, 상기 단위동작은 처음과 끝의 포즈가 일치하는 것이어도 좋다.

<58> 바람직하게는, 상기 단위동작의 진행시간은 빠르기에 의해 조절되는 것이어도 좋다.

<59> 바람직하게는, 상기 단위동작은 전, 후, 좌, 우, 및 전-좌, 전-우, 후-좌, 후-우측의 8방향으로의 이동을 포함하는 것이어도 좋다.

<60> 바람직하게는, 상기 단위동작은 90도 회전, 180도 회전, 360도 회전 및 특수 단위동작을 포함하는 것이어도 좋다.

<61> 바람직하게는, 상기 단위동작은 앉기, 서기, 구부리기, 연속회전을 포함하는 것이어도 좋다.

<62> 바람직하게는, 상기 단위동작은 구조체를 구성하고 있는 관절 및 이에 의한 동작변형을 포함하는 것이어도 좋다.

<63> 바람직하게는, 상기 단위동작은 구조체를 구성하고 있는 관절 및 이에 의한 다수의 동작의 결합을 포함하는 여러 개를 하나의 단위로 하는 것이어도 좋다.

<64> 바람직하게는, 상기 이벤트는 키보드, 마우스, 조이스틱, 키패널 중 적어도 하나를 이용하여 입력하는 것이어도 좋다.

<65> 바람직하게는, 상기 이벤트는 각종 센서 또는 카메라를 통해 위치값을 입력하여 동작데이터를 입력하는 것이어도 좋다.

<66> 바람직하게는, 상기 구조체의 동작에 대한 조종시, 조종기기의 기계적 제어에 의한 시간적 효과나 항력과 작용/반작용과 같은 공간적, 물리적 효과를 포함하여 처리하는 것이어도 좋다.

<67> 바람직하게는, 상기 구조체는 2차원 또는 3차원 오브젝트인 것이어도 좋다.

<68> 바람직하게는, 상기 구조체는 별도의 모델도구를 통해 만들어진 아바타인 것이어도 좋다.

<69> 바람직하게는, 상기 오브젝트는 카메라 등을 통해 입력되는 영상을 바탕으로 제작된 오브젝트와 실제 영상과의 결합을 통해 구현되는 것이어도 좋다.

<70> 바람직하게는, 상기 클라이언트 시스템은 별도의 채팅툴을 포함하여, 상대방 클라이언트 시스템과 문자 혹은 음성으로 의사를 교환하는 것이어도 좋다.

<71> 바람직하게는, 상기 클라이언트 시스템은 별도의 메모리를 가지고 온라인 혹은 2인 이상 게임이 가능한 PS2 나 X-box, GameCube와 같은 비디오 게임기를 포함하는 것이어도 좋다.

<72> 바람직하게는, 상기 단위동작은 스포츠댄스와 같이 2인이 함께 붙어서 진행되는 것이어도 좋다.

<73> 바람직하게는, 스포츠 댄스는 왈츠, 탱고, 폭스트로트, 비엔나 왈츠, 퀵스텝, 자이브, 룸바, 차차차, 삼바, 파도소블, 블루스 중 하나 또는 하나 이상의 결합에 의한 것이어도 좋다.

<74> 이하에서, 첨부된 도면을 참조하면서 이 발명의 한 실시 예에 따른 클라이언트 시스템의 동작 동기화 장치와 방법 및 이를 이용한 통신망을 통한 상호작용 시스템 및 게임방법을 보다 상세하게 설명한다.

<75> 도 1은 이 발명의 한 실시 예에 따른 동작 동기화 장치를 포함한 클라이언트 시스템의 개략적인 구성도이다.

<76> 이 발명에 따른 클라이언트 시스템은 입력부(100)와 연산처리부(110)와 동기화부(120)와 인터페이스부(130)와 출력부(140)를 포함한다.

<77> 입력부(100)는 사용자의 요구에 따라 이벤트를 발생시킨다. 이 입력부(100)는 휴먼 인터페이스로서 컴퓨터 혹은 휴대장치에서 다수의 키에 의해 데이터 및 정보를 이 발명에 따른 동작 동기화 장치에 입력한다. 통상적으로 입력부(100)는 키보드, 마우스, 트랙 볼, 조이스틱, 터치스크린 및 핸드폰 키패드 중 하나 또는 다수의 조합으로 구현된다.

<78> 연산처리부(110)는 도시되지 않은 중앙연산장치(Central Processing Unit)와, 롬(ROM)과 램(RAM)과 CRT제어부와 제어부를 포함한다.

<79> 여기서, 중앙연산장치(CPU)는 제어프로그램에 의하여 연산 및 시스템 제어를 수행하며, 마이크로프로세싱유닛(MPU) 등으로 이루어지고 롬에 저장된 제어프로그램을 기동시키며, 그 제어프로그램에 따라 데이터 제어 처리를 실행하는 연산을 수행한다.

<80> 그리고, 롬(ROM)은 비휘발성 메모리로서, 중앙연산장치의 제어프로그램을 저장한다.

<81> 그리고, 램(RAM)은 중앙연산장치가 제어프로그램를 구동하는데 필요한 데이터나 컨텐츠 또는 중앙연산장치의 연산과정에서 필요한 연산결과를 저장한다.

<82> 그리고, CRT제어부는 램(RAM)에 저장되어 있는 데이터나 컨텐츠를 어드레스를 통해 소정 주기에 따라 순차 판독하여 화상신호로 변환하여 출력부(140)에게 출력한다.

<83> 그리고, 제어부는 CRT제어부에서 발생한 화상신호와 음성신호를 각각 인터페이스부(130)를 통해 화면 출력부와 음원 출력부에게 전달한다.

<84> 인터페이스부(130)는 연산처리부 내부의 중앙연산장치와 롬과 램과 CRT제어부를 외부장치인 입력부와 메모리부와 표시부와 정합한다.

<85> 동기화부(120)는 입력부(100)에서 발생한 이벤트를 표준시간과 비교하여 시간차를 보정함으로써, 사용자간의 시간격차를 동일하게 일치시키는 역할을 한다. 이 동기화부(120)는 도시되지 않은 시간설정부와 시간동기부와 단위동작설정부와 단위동작동기부를 포함하여 구성된다.

<86> 여기서, 시간설정부는 사용자의 시간을 하나의 표준으로 동일하게 일치시키기 위한 표준시간을 설정한다. 표준시간을 설정하는 방법으로는 온라인이나 인터넷이 연결된 상태에서는, 인터넷상의 원자 시계 서버 혹은 온라인이나 웹 서비스를 위한 서버의 시계를 표준시간으로 설정하고 사용자의 시간을 표준시간에 일치시키는 방법을 사용한다. 온라인 및 웹이 연결되지

않은 상태에서 장치간의 연결된 경우에는 각각의 시간 혹은 장치에 지정된 시간을 그대로 표준 시간으로 사용할 수도 있다.

<87> 그리고, 시간동기부는 시간설정부에서 설정된 표준시간과 사용자 간의 전송시간을 계산하여 서버에서 정보를 전송할 경우 발생하는 전송시간의 차이에 따른 시간차를 계산하고 이를 사용자의 인터페이스부를 통해 전달하여 줌으로써 정보 전송에 따른 이벤트의 시간차를 제거하는 역할을 한다. 또한, 서버의 부하에 따른 전송시간의 지연을 고려하여 시간차를 재계산하여 이벤트를 일치시키는 역할도 함께 수행한다.

<88> 그리고, 단위동작설정부는 구조체를 구성하고 있는 관절 및 이에 의한 동작에 의한 단위동작을 구분하여 진행, 회전, 평형유지와 같은 동작을 진행할 수 있도록 한다. 이러한 단위동작설정부에서 다양한 동작의 진행에 따른 데이터의 생성방식으로는, 보간에 의해 계산하는 방식과 스케레탈 애니메이션(Skeletal Animation)과 골격 애니메이션(Bone Animation 또는 Skining Animation) 방식이 있다. 보간에 의한 방식(Vertex Animation 또는 Key Frame Animation)은 각각의 동작에 의한 장면에 해당되는 프레임에서 각 정점의 위치와 관련 데이터를 저장하고 있다가 이것을 선형보간이나 그 이외의 보간에 의해 계산하는 방식이다. 스케레탈 애니메이션은 구조체를 여러 개로 나누어서 각각의 관계를 규정하고, 나누어진 구조체마다 이동, 축소, 회전을 포함한 데이터를 매 프레임 또는 변화되는 프레임마다 저장하여 사용하는 방식이다. 골격 애니메이션은 골격(bone)이라는 관절 단위의 구조체 데이터를 토대로 각각의 관계를 규정한 계층적 구조를 가지고 위치값을 가지고 움직이는 방식이다. 골격 애니메이션 방식을 사용하면 부드러운 동작과 적은 데이터 파일로도 동작에 따른 데이터 생성이 가능하다.

<89> 그리고, 단위동작동기부는 사용자간에 원격으로 접속되어 있는 하나의 구조체와 다른 구조체와의 상호간섭을 고려하기 위해, 설정된 표준시간을 바탕으로 각

사용자에게 동기화된 시간을 토대로 구조체간의 이벤트 및 동작발생의 동시성을 구현하도록 한다. 이러한 단위동작동기부에서 단위동작을 맞추기 위해서 필요에 따라 부수적으로 음원을 사용하는 것이 가능하며, 이때 사용하는 음원으로서 WAV, MP3, WMA, MIDI 및 이와 유사한 형태의 음원을 사용한다. 이 경우 동작진행과 음원의 박자를 일치시킬 수도 있고, 음원에서 시간 진행에 맞추어 표준시간을 설정하고 이를 단위동작과 일치시키는 것도 가능하다.

<90> 출력부(140)는 화면 및 음원을 출력한다. 컴퓨터 및 휴대장치에서 송수신되는 데이터 및 제어여부를 모니터, HUD(Head UP Display device)장치, LCD 패널 등과 같은 화상 표시소자의 화면에 표시하거나, 음원신호를 스피커 등을 통해 출력하여 입출력 매개상태를 확인하도록 한다. 또는 화면이나 음원을 출력하는 별도 기기와의 연결을 통하여 입출력 매개 상태를 확인하는 것도 가능하다.

<91> 상기와 같이 구성된 이 발명에 따른 동작 동기화 장치의 작용을 설명한다.

<92> 먼저, 동기화부는 상술한 방법들 중 한가지 방법을 이용하여 표준시간을 설정한다.

<93> 그리고, 단위시간동안 움직이는 단위동작을 설정한다. 앞으로 걷기 동작에 있어서, 임의의 단위시간의 처음과 마지막 자세는 기본자세이고, 단위시간동안에 걷기에 대한 단위동작이 실행된다.

<94> 예컨대 사용자가 걷기 동작 이벤트를 입력하면, 연산처리부는 이벤트를 입력한 순간의 다음 단위시간에 동기되어 구조체가 해당 단위시간 동안에 걷기 동작을 하도록 출력한다. 그리고, 구조체가 걷기 동작을 수행하는 단위시간내의 임의의 순간에 사용자가 입력한 이벤트는 그 다음 단위시간 동안에 구조체에 적용되어 실행된다.

<95> 도 2는 이 발명의 한 실시 예에 따른 동작 동기화 장치의 기능 블록도이다.

<96> 이러한 동작 동기화 장치는, 도 1에서도 언급한 바와 같이, 입력부(100)와 연산처리부(110)와 동기화부(120)와 인터페이스부(130)와 출력부(140)를 포함한다.

<97> 입력부(100)는 입력기(101)를 구비하는데, 입력기(101)는 키보드, 마우스, 조이스틱, 키패널과 같은 휴먼 인터페이스로서, 컴퓨터 혹은 휴대장치에서 다수의 키에 의해 이벤트를 발생시킨다. 또한, 입력기(101)는 카메라나 센서를 통한 동작의 직접입력 혹은 마이크와 같은 음성입력을 통한 명령 입력과 같은 형태도 가능하다.

<98> 동기화부(120)는, DB처리기(121)와 가상공간처리기(122)와 개인정보처리기(123)를 구비한다.

<99> 여기서, DB처리기(121)는 컴퓨터 혹은 휴대장치에 포함된 혹은 별도로 추가 가능한 형태로써 사용자의 사용기록이나, 점수, 레벨 등과 같은 각 사용자의 기록을 데이터베이스화하여 본 시스템에서 필요시 불러올 수 있도록 한다.

<100> 그리고, 가상공간처리기(122)는 시스템의 기동시 가상공간을 연산처리부(110)의 램에 적재하고, 사용 중에는 사용자의 이벤트 입력에 따라 공간을 조정해 주며, 시스템의 종료시 가상공간을 램에서 소멸시킨다.

<101> 그리고, 개인정보처리기(123)는 시스템 기동시 사용자의 개인정보를 인증하고, 종료시 개인 정보를 저장하는 작업을 처리하기 위한 모듈로서, 호스트 서버와의 네트워크망 통신을 이용해서 처리된다.

<102> 연산처리부(110)는 이벤트처리기(111)와 액세스데이터처리기(112)와 GUI(Graphic User Interface)처리기(113)를 구비한다.

<103> 여기서, 이벤트처리기(111)는 GUI처리기(113)의 진행에 따라 입력기(101)를 이용한 사용자의 입력데이터를 액세스 데이터로 변환시킨다.

<104> 그리고, 액세스데이터처리기(112)는 액세스데이터전송기와 액세스데이터판정기로 구성되며, 액세스데이터전송기는 이벤트처리기(111)가 생성한 액세스데이터를 타 클라이언트에게 보내고, 타 클라이언트로부터 액세스데이터를 받아들이는 역할을 한다. 액세스데이터판정기는 주어진 조건인 시간과 요구된 데이터를 비교 판단하는 역할을 하며, 그 결과를 인터페이스부(130)에게 보낸다.

<105> 그리고, GUI처리기(113)는 출력부(140)를 통하여 사용자에게 현재 시스템이 진행되는 각종 상황을 모니터할 수 있게 하며, 또한 사용자가 이벤트를 생성해야 할 시점을 지시하는 역할을 한다.

<106> 인터페이스부(130)는 동작진행처리기(131)를 구비하며, 동작진행처리기(131)는 액세스데이터처리기(112)로부터 전달된 정보를 이용하여 동기화부(120)에서 설정된 단위동작을 각 클라이언트에서 실행시킨다.

<107> 출력부(140)는 출력기(141)를 구비하며, 출력기(141)는 동작진행처리기(131)에 의해 처리된 단위동작을 화면 및 음원을 동반하여 출력한다.

<108> 도 3은 이 발명의 한 실시 예에 따른 동작 동기화 장치를 이용한 통신망을 통한 상호작용 시스템을 도시한 기능 블록도이다.

<109> 통신망을 통한 상호작용 시스템은, 도 3에 도시된 바와 같이, 다수의 클라이언트 시스템(310, 320)과 호스트서버(330)가 결합된 서버/클라이언트(Server/Client, S/C)방식으로 구현될 수도 있고, 피어투피어(Peer to Peer, P2P)방식으로 구현될 수도 있다. 또는 별도의

메모리를 가지고 온라인 혹은 2인 이상 게임이 가능한 PS2 나 X-box, GameCube와 같은 비디오 게임기 방식으로 구현될 수도 있다.

<110> 이 S/C방식과 P2P방식의 차이는 접속되는 사용자의 수에 기인하는데, 대규모의 사용자가 접속해야 할 필요가 있을 경우 S/C방식을, 소수의 사용자만을 고려하는 경우에는 P2P방식을 선택적으로 적용한다.

<111> S/C 방식에서 호스트서버는 데이터베이스를 관리하여 각 클라이언트 시스템의 정보처리를 통하여 개인간의 정보를 처리한다. 호스트서버의 데이터베이스를 토대로 각 원격 클라이언트 시스템간에 이벤트 처리가 가능해지고 액세스데이터처리를 통하여 상호간 액세스데이터가 공유된다. 상기 도 3에는 하나의 호스트 서버에 2대의 클라이언트만이 연결된 것으로 되어 있으나, 더 많은 클라이언트 시스템의 연결이 가능하다.

<112> 한편, P2P방식은 호스트서버 없이 두 클라이언트 시스템이 상호간에 개인정보를 처리한 후 서로 공유한다. 이러한 P2P방식은 정보공유형과 자원공유형 모두에 대한 실시간 통신이나 자원분배가 가능하며, 핑(Ping), 퐁(Pong), 쿼리(Query), 쿼리히트(Queryhit), 푸쉬(Push) 등과 같은 스트립터의 사용이 자유롭게 된다. 또한 PS2 나 X-box, GameCube와 같은 비디오 게임기는 게임기에 별도의 메모리가 있으므로 여기에서 데이터베이스를 관리하고 상호간 액세스 데이터를 공유할 수가 있다.

<113> 각각의 클라이언트 시스템은, 도 2에 도시된 바와 같은, 동작 동기화 장치를 포함하며, 입력기로부터 입력되는 이벤트에 의해 움직이는 구조체와 타 클라이언트 시스템으로부터 입력되는 이벤트에 의해 움직이는 구조체를 포함한다.

<114> 즉, 도 3에 도시된 바와 같이, 2개의 클라이언트 시스템이 연결된 경우, 각각의 클라이언트 시스템의 출력부 화면에는 2개의 구조체가 표시되는데, 하나는 자신의 입력기로부터 입력되는 사용자의 이벤트에 의해 움직이고 다른 하나는 상대방 클라이언트 시스템으로부터 입력되는 이벤트에 의해 움직인다.

<115> 두 클라이언트 시스템은 표준시간으로 동기가 맞춰지고, 임의의 기준시간동안 입력된 이벤트에 의해 다음 기준시간동안 해당 이벤트에 따른 단위동작이 실행된다.

<116> 도 4 내지 도 6은 이 발명의 한 실시 예에 따른 상호작용 시스템을 이용한 댄스 게임방법을 도시한 동작 흐름도이고, 도 7 내지 도 9는 상기 댄스 게임방법이 진행되는 동안에 디스플레이되는 예시 화면이다.

<117> 설명의 편의상 도 4 내지 도 6의 게임방법은 P2P방식으로 구현된 상호작용 시스템에 적용되는 인터렉티브 댄스 게임방법이라고 가정한다.

<118> 제1게임자는 제1클라이언트 시스템을, 제2게임자는 제2클라이언트 시스템을 각각 이용하여 게임하는 것으로 가정하고, 제1게임자가 리더(leader), 제2게임자가 팔로우어(follower)라고 가정한다.

<119> 표준시간을 다수의 기준시간구간(World Time Code: WTC)으로 나누었을 때, 각 게임자들은 임의의 기준시간구간 중에 동작 이벤트를 입력하고, 그 입력된 동작 이벤트는 통신망을 이용하여 상대방 클라이언트 시스템에게 전송되며, 그 동작 이벤트에 해당하는 동작이 다음 기준시간구간(WTC) 동안에 화면에 표시된다.

<120> 게임자에 의해 직접 입력되는 동작이벤트를 로컬스텝신호라고 하고, 상대방 클라이언트 시스템으로부터 입력되는 동작이벤트를 원격스텝신호라고 한다.

<121> 즉, 제1클라이언트 시스템의 경우, 제1게임자에 의해 입력되는 동작이벤트를 로컬스텝신호라고 하고 제2클라이언트 시스템로부터 입력되는 동작이벤트를 원격스텝신호라고 한다. 제2클라이언트 시스템의 경우, 제2게임자에 의해 직접 입력되는 동작이벤트를 로컬스텝신호라고 하고 제1클라이언트 시스템으로부터 입력되는 동작이벤트를 원격스텝신호라고 한다.

<122> 도 7은 사용자의 모니터에 표시되는 GUI화면의 일 예시도이다. 사용자 GUI화면은 춤을 추기 위한 무대배경과 스텝신호를 표시하는 스텝이미지(71), 화면중심부의 구조체(캐릭터)(72), 단위동작 입력 실패누적수 또는 성공누적수(73) 및 시간게이지(74)가 표시된다. 이 시간게이지(74)는 단위동작 입력 가능시간으로서, 현재 WTC 중 경과한 시간과 남은 시간을 표시한다.

<123> 도 4는 클라이언트 시스템에서의 댄스 게임방법을 도시한 동작 흐름도이다.

<124> 먼저, 게임이 시작되면 상대방 클라이언트 시스템과 동기를 맞춘다(S401).

<125> 도 8a 내지 도 8d 및 도 9a 내지 도 9d는 동기를 맞추는 과정의 일 예시 화면이다.

<126> 먼저, 도 8a 및 도 9a와 같이 각 클라이언트 시스템에 게임자들이 게임시작을 확인하는 대화창이 표시된다. 이 대화창에서 각 게임자들이 확인버튼을 클릭하면, 도 8b 및 도 9b와 같이 배경화면과 구조체가 표시되면서 상대방 클라이언트 시스템으로부터 동기신호가 수신되기를 대기한다. 쌍방간에 동기신호가 수신되면 도 8c 및 도 9c와 같이 댄스 게임 서비스 준비상태로 전환된다.

<127> 그리고, 도 8d 및 도 9d와 같이 댄스 게임이 시작되어 임의의 한 기준시간구간(WTC_i)이 시작되면(S402), 화면에 동작입력 준비중 상태임을 표시한다(S403). WTC_i이 시작될 때에는

모두 남은 시간으로 표시되고 시간이 경과할수록 경과한 시간은 증가하고 남은 시간은 감소하도록 표시된다.

<128> 단위동작 입력 가능시간동안에 동작 입력 처리 서브루틴(S404)을 수행한다. 이 동작 입력 처리 서브루틴이 수행되는 동안에 각 게임자는 동작이벤트를 입력하는데, 리더인 제1게임자가 먼저 동작이벤트를 입력하면 제1게임자가 입력한 스텝신호가 제1 및 제2클라이언트 시스템에게 표시되고, 제2게임자는 화면에 표시된 스텝이미지를 보고 동작이벤트를 입력하면 제2게임자가 입력한 스텝신호가 제1클라이언트 시스템에게 전달된다. 이 동작 입력 처리 서브루틴의 상세한 설명은 도 5와 도 6을 참조하면서 후술하기로 한다.

<129> 즉, 동작 입력 처리 서브루틴(S404)이 정상적으로 수행된 후에는 각 클라이언트 시스템은 게임자에 의해 직접 입력된 동작이벤트에 따른 로컬스텝신호와 상대방 클라이언트 시스템으로부터 입력된 원격스텝신호가 존재하게 된다.

<130> 동작 입력 처리 서브루틴(S404)후 두 게임자가 입력한 동작이벤트가 일치하는 지를 검사한다(S405).

<131> 단계 S405의 검사 결과, 두 게임자가 입력한 동작이벤트가 일치하면 도 8f 및 도 9f와 같이 동작입력 성공 메시지를 화면에 표시하고(S406), 입력 성공한 동작에 대한 랜더링을 준비한다(S407).

<132> 한편, 단계 S405의 검사 결과, 두 게임자가 입력한 동작이벤트가 일치하지 않으면 도 8g 및 도 9g와 같이 동작입력 실패 메시지를 화면에 표시하고(S408), 입력 실패한 동작에 대한 랜더링을 준비한다(S409). 이 입력된 동작에 대한 랜더링은 WTC_i+1 구간에 처리된다.

<133> WTC_i 구간이 종료하지 않았으면(S410), 동작입력 실패한 경우에 동작 입력 처리 서브루틴(S404)을 다시 실행하도록 할 수 있다.

<134> WTC_i 구간이 종료되면(S410), 동작입력의 성공여부를 판단하여(S411), 성공한 경우에는 성공횟수를 누적하고 성공동작 랜더링을 수행하며(S412), 실패한 경우에는 실패횟수를 누적하고 실패동작 랜더링을 수행한다(S413).

<135> 그리고, 게임종료여부를 판단하여 게임이 종료되지 않았으면 i를 1 증가시킨 후(S415), 단계 S403으로 되돌아간다. 엄밀하게 말하면 단계 S412와 S413의 성공동작 랜더링과 실패동작 랜더링은 WTC_i+1 구간에서의 단계 S403 내지 단계 S409가 처리되는 동안에 처리되나, 본 실시예에서는 이 발명의 이해를 돕고자 별도로 표시한다.

<136> 상술한 댄스 게임은 댄스 서비스의 배경음악이 끝나거나 실패누적횟수가 정해진 횟수를 초과한 때 종료한다. 실패누적횟수를 초과하지 않은 상태에서 배경음악이 끝나면 도 8h 및 도 9h와 같이 댄스게임 클리어화면이 출력된다.

<137> 도 5는 리더인 제1클라이언트 시스템에서의 동작 입력 처리 서브루틴을 도시한 동작 흐름도이다.

<138> 제 1게임자로부터 키입력이 있으면(S501), 해당 키입력이 정상적인 동작이벤트인지를 검사한다(S502).

<139> 정상적인 동작이벤트가 아니면 동작입력 실패 메시지를 화면에 표시하고(S503), 단계 S501로 되돌아간다.

<140> 단계 S502에서 정상적인 동작이벤트가 입력된 것이면, 제1클라이언트 시스템은 로컬스텝 신호를 생성하고(S504), 도 8e와 같이 해당 로컬스텝신호를 화면에 표시하며(S505), 아울러 로

컬스텝신호를 상대방인 제2클라이언트 시스템에게 전송한다(S506). 그리고, 상대방인 제2클라이언트 시스템으로부터 원격스텝신호를 수신하면 단계 S405로 복귀한다(S507).

<141> 도 6은 팔로우어인 제2클라이언트 시스템에서의 동작 입력 처리 서브루틴을 도시한 동작 흐름도이다.

<142> 상대방인 제1클라이언트 시스템으로부터 원격스텝신호가 수신되면(S601), 도 9e와 같이 수신된 원격스텝신호를 화면에 표시하고(S602), 제2게임자로부터의 키입력을 대기한다.

<143> 키입력을 있으면(S603), 해당 키입력이 정상적인 동작이벤트인지를 검사한다(S604). 정상적인 동작이벤트이면 로컬스텝신호를 생성하고(S605), 생성된 로컬스텝신호를 상대방인 제1클라이언트 시스템에게 전송한 후 단계 S405로 복귀한다(S606). 한편, 단계 S604에서 정상적인 동작이벤트가 아니면 동작입력실패메시지를 화면에 표시하고(S607), 단계 S603으로 되돌아간다.

<144> 이 발명에 따른 댄스 게임의 경우, 두 구조체(캐릭터)가 서로 마주보는 형태로 결합하고, 게임자는 전, 후, 좌, 우 및 전-좌, 전-우, 후-좌, 후-우측의 8방향의 단위동작으로 구조체를 이동시키고, 이외에 90도 회전, 180도 회전, 360도 회전 및 별도의 특수 단위동작을 포함하여 12개의 단위동작을 가지도록 한다. 하나의 단위동작을 수행하는데 소요되는 시간은 WTC 1단위에서 WTC 4단위까지이고, 각 단위동작의 연결 포즈를 부드럽게 하기 위해서 단위동작의 처음과 끝의 포즈가 반드시 정확히 일치하게 설정한다. 여기에서 단위동작은 이해를 돕기 위해 방향에 대해서만 설명하였으나 하나의 단위동작 내에서 구조체를 구성하고 있는 관절 및 이에 의한 동작변형(예를 들어 면 한쪽 팔을 위로 높이 들거나, 위에서 둥글게 흔들거나, 팔을 구부렸다가 펴거나, 혹은 팔을 올렸다가 내리는 등과 같은 다수의 결합, 혹은 방향과는 별도로

키의 입력에 따라 방향 및 팔 다리의 움직임에 따른 다수의 동작의 결합을 포함하는) 여러 개를 하나의 단위로 포함시킬 수 있다.

<145> 게임이 시작되면 배경음악의 진행시간 경과에 따라 제1클라이언트 시스템(리더)은 제2클라이언트 시스템(팔로우어)에게 WTC_i 동기신호를 전송하고, 각 클라이언트 시스템은 기준시간구간 WTC_i의 시간동안 제1게임자로부터 입력된 동작이벤트에 해당하는 스텝이미지를 속이 빈 상태로 표시한다.

<146> 기준시간구간 이후 제1게임자와 제2게임자에 의해 입력된 동작이벤트가 일치하면 속이 찬 상태로 표시하고, 다음 기준시간구간인 WTC_i+1동안에 입력 성공에 따른 동작이 진행되거나, 포인트를 가산하여 동작의 연결이 성공적으로 이루어졌음을 표시할 수 있으며, 필요에 따라 화면상에 이를 파란색의 램프나 혹은 OK, 성공과 같은 표시를 할 수도 있다.

<147> 그러나, 기준시간구간 이후 제1게임자와 제2게임자에 의해 입력된 동작이 일치하지 않으면 다음 기준시간구간인 WTC_i+1동안 입력실패에 따른 동작이 진행 되고, 출력부에 단위동작 입력 실패를 표시한다.

<148> 이와 같은 동작 입력의 불일치가 발생 또는 누적될 경우, 게임의 진행이 중단되거나 혹은 포인트의 감소, 또는 빨간색의 램프표시나 FAIL, 실패와 같은 메시지를 화면에 출력할 수도 있다. 또는, 화면 출력이 아닌 음향이나 음성을 통한 출력도 가능하다.

<149> 단위동작의 마지막 프레임에서 구조체는 마지막 포즈의 위치로 이동하고 다음 단위동작을 시작한다. 이것은 가상공간에서 모델 등(오브젝트)의 이동을 속도벡터로 처리할 경우 오브젝트가 일정 속도와 일정 방향으로 이동하기 때문에 춤 동작이나 격투동작 등의 모델이 취하는

개별동작(발이 닿는 위치 등)을 가상공간에 매치시킬 수 없는 문제점을 해결하기 위한 것이다.

<150> 이 발명의 실시 예에서는 춤 단위동작의 처음과 끝의 포즈가 반드시 정확히 일치한 것으로 설명하였으나, 이는 하나의 예일 뿐이며 처음과 끝의 포즈를 일치시키지 않을 수도 있다. 또한, 단위동작의 진행시간은 필요에 따라서 템포(빠르기)를 조절하는 것이 가능하다.

<151> 또한 이 발명의 실시 예에서는 추가적으로 일련의 스텝이미지들을 게임자의 화면에 표시하여 일정 형식의 댄스를 구현하는 것도 가능하다.

<152> 또한 이 발명의 실시 예에서 단위동작을 12개로 한정하였으나 댄스의 종류에 따라 상기 12개의 단위동작 이외에도 앉기, 서기, 구부리기, 연속 회전과 같은 형태로 동작을 추가하거나 변경하는 것이 가능하다.

<153> 또한, 입력부에서는 키보드나 마우스, 조이스틱, 키패널과 같은 장치를 통해 입력하는 것을 설명하였으나 가상현실과의 접목을 위한 각종 부착 센서, 혹은 카메라를 이용하여 위치값을 구하는 방법 혹은 조합이나 병행 등을 통하여 대상물에서 동작 데이터를 입력하는 것이 가능하다.

<154> 또한 클라이언트에서 사용되는 모델은 통상의 2차원 혹은 3차원 오브젝트를 그 대상으로 사용하지만 별도의 모델도구를 통해 만들어지는 아바타(Avatar) 혹은 카메라 등을 통해 입력되는 영상을 토대로 제작되는 오브젝트와 실제영상과의 결합을 통해 구현되는 것도 가능하다.

<155> 또한 입력부와는 별도로 매신저와 같은 채팅툴을 사용하여 상대방과의 문자 혹은 음성을 통해 의사를 교환하는 것이 가능하다.

<156> 또한 출력부는 모니터 등을 통하여 동작이 출력되는 것에 대해 설명하였으나 입체로 구성된 대상물(예를 들면, 사람을 포함하여 동물, 로보트, 비행기 등)과의 유선, 혹은 무선 상의 송수신을 통하여 하나 혹은 하나 이상이 결합된 대상물을 조정하는 것도 가능하다.

<157> 이 발명은 상기한 실시 예에 한정되지 않고, 여러 가지의 형태로 실시할 수도 있다.

<158> 즉, 2인 3각이나 2인 이상이 하나의 구조체를 형성하여 조작하여야 하는 다양한 응용 게임에도 적용 가능하다. 예를 들어, 중국의 사자놀이과 같이 구조체의 단위동작과 동기화가 필요하며 평형유지가 요구되는 게임이나, 써커스나 곡예공연에서 볼 수 있는 2인 이상의 줄타기, 공굴리기, 인간 탑 쌓기와 같은 균형이 요구되는 동작을 응용한 게임, 혹은 1인용 시뮬레이션 게임에서 구조체의 동작에 대한 대상물체 동작조종 시, 조종기기의 기계적 제어에 의한 시간적 효과나 공간적 효과(즉, 공기 중, 수중, 우주에서의 항력과 작용, 반작용에 따른 효과 등)를 단순한 시간지연으로 처리하지 않고 물리적 영향을 포함하여 처리하는 경우 등에 응용하여 적용할 수 있다.

<159> 또한 스포츠댄스와 같이 2인이 함께 붙어서 진행되는 게임의 경우도 가능하다. 스포츠댄스는 왈츠, 탱고, 폭스트로트, 비엔나 왈츠, 퀵스텝과 같은 모던 5종 이외에 자이브, 룸바, 차차차, 삼바, 파도소블과 같은 라틴 5종목이나 블루스등과 같은 동작으로도 진행이 가능하다.

<160> 또한 본 실시 예에서는 게임자가 2인인 경우에 대하여 설명하였으나, 이 발명은 이에 한하지 않고 게임자가 하나 이상으로 구성되는 다수의 구성으로 하여도 무방하다.

<161> 마찬가지로, 본 실시 예에서는 구조체 2개가 상호간섭에 의하여 하나의 형태로 진행되는 경우에 대하여 설명하였으나 이 발명에 따르면 하나 이상의 형태로 구성되는 다수의 경우에 대해서도 동일하게 적용할 수 있다.

<162> 상기에서 설명한 이 발명에 관한 소스 프로그램의 예를 들면 다음과 같다.

```
<163> //------------------------
<164> // part D - after connect Game Windows display
<165>    if( g_b_isConnectSuccess )
<166> {
<167> int iCount = 0;
<168> // App is now connected via DirectPlay, so start the game.
<169> g_hrResult = S_OK;
<170> //-----------------------------
<171> g_b_isLeader = g_pNetStage->IsLeaderPlayer();
<172> g_pRemote->b_isLeader_local = g_b_isLeader;
<173> //myLog_Printf("Wn==============================Wn");
<174> myLog_Printf("base at : b_isLeader_local : %dWn", g_pRemote->b_isLeader_local);
<175> //-------------------------------
<176> // Game application ceate
<177> // create multiplay two player
<178> pM2p = M2P_Create(
<179> g_app_hInst, g_app_hWnd,
<180> Preset->Camera,
<181> Preset->Engine, Preset->i_Width, Preset->i_Height,
```

```
<182> DebugPath,
<183> g_pRemote
<184> );


<185> if (pM2p != NULL) pM2p->b_isAfterCreate = TRUE;
<186> else MessageBox(NULL,"Failed to create a M2P structure!!","Error",MB_OK), exit(-1);
<187> //----------------------
<188> // after create signal
<189> // trans game data : after_create
<190> if(pM2p->b_isAfterCreate)
<191> {
<192> {
<193> HRESULT hr = S_OK;
<194> hr = SendTo_GameData(GAME_MSGID_AFTER_CREATE, (LONG)pM2p->b_isAfterCreate, TRUE);
<195> if( FAILED(hr) )
<196> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<197> }
<198> }
<199> f_OldTimeGetTime = (float) timeGetTime();
<200> while (b_isWaiting)
```

```
<201> {
<202> if(g_pRemote->b_isAfterCreate)
<203> {
<204> b_isRunning = JE_TRUE;
<205> b_isWaiting = FALSE;
<206> }
<207> //for exit this loop
<208> if ( IsKeyDown( VK_ESCAPE ) )
<209> {
<210> //jeEngine_Printf(pM2p->Engine, 100, 100, "have been pressed ESC key");
<211> b_isWaiting = FALSE;
<212> goto END_LOOP;
<213> }
<214> //---------------
<215> if(PeekMessage(&Msg, NULL, 0, 0, PM_REMOVE))    //non-blocking message check
<216> {
<217> // stop after quit message is posted
<218> if (Msg.message == WM_QUIT)
<219> // break;       // <---loop ends here
<220> goto END_LOOP;
```

```
<221> TranslateMessage(&Msg);
<222> DispatchMessage(&Msg);
<223> }
<224> }
<225> myLog_Printf("COMPLETE SYNCRONIZED STEP1 ==========> Wn");
<226> //--------------------------------
<227> f_OldTimeGetTime = (float) timeGetTime();
<228> while (b_isRunning)
<229> {
<230> //---------------
<231> // Update the application
<232> if (!App_Update()) b_isRunning = FALSE;
<233> if (!App_Render()) b_isRunning = FALSE;
<234> if (!App_UserInput()) b_isRunning = FALSE;
<235> //---------------
<236> if(PeekMessage(&Msg, NULL, 0, 0, PM_REMOVE))    //non-blocking message check
<237> {
<238> // stop after quit message is posted
<239> if (Msg.message == WM_QUIT)
<240> break;       // <---loop ends here
```

```
<241> TranslateMessage(&Msg);
<242> DispatchMessage(&Msg);
<243> }
<244> }
<245> END_LOOP:
<246> //-----------------------------------
<247> // for multiplay two player
<248> if(!M2P_Shutdown(pM2p))
<249> MessageBox(NULL, "Failed M2P_Shutdown()!!","Error", MB_OK|MB_ICONERROR), _exit(-1);
<250> if(!myLog_Report("Debug_data_trans_test.txt"))
<251> MessageBox(NULL, "Failed myLog_Report()!!","Error", MB_OK|MB_ICONERROR), _exit(-1);
<252> //  goto REENTRY_STAGE;
<253> }
<254> //-----------------------------
<255> if( FAILED( g_hrResult ) )
<256> {
<257> if( g_hrResult == DPNERR_CONNECTIONLOST )
<258> {
<259> MessageBox( NULL, TEXT("The DirectPlay session was lost. ")
<260> TEXT("The test will now quit."),
```

```
<261> TEXT("Dance P2P Test"), MB_OK | MB_ICONERROR );
<262> }
<263> else
<264> {
<265> DXTRACE_ERR( TEXT("DialogBox"), g_hrResult );
<266> MessageBox( NULL, TEXT("An error occured during the game. ")
<267> TEXT("The test will now quit."),
<268> TEXT("Dance P2P Test"), MB_OK | MB_ICONERROR );
<269> }
<270> goto QUIT_GAME;
<271> }
<272> ////////////////////////////////////////////////////////////////////////////////////////////////
     ////////////////////////////////////////////////

<273> BOOL App_Update()
<274> {
<275> // local
<276> float f_current_time;
<277> BOOL b_isWaiting = TRUE;
<278> int iCount = 0;
```

```
<279> assert(Preset);
<280> //ENGINE----------------------------------------------------------------
<281> //turn on the engine
<282> if (!jeEngine_Activate(Preset->Engine, JE_TRUE)) MessageBox(g_app_hWnd, "Engine did not
      activate", "Debug", MB_OK);
<283> //-------------------------
<284> // trans game data : befor_update
<285> pM2p->b_isBeforeUpdate = TRUE;
<286> if (!g_b_isFirstEntryToUpdate)
<287> {
<288> if(pM2p->b_isBeforeUpdate)// && g_b_isSyncStep1)
<289> {
<290> {
<291> HRESULT hr = S_OK;
<292> hr = SendTo_GameData(GAME_MSGID_BEFORE_UPDATE, (LONG)pM2p->b_isBeforeUpdate, TRUE);
<293> if( FAILED(hr) )
<294> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<295> }
<296> //   for ( i = 0; i < 10000; i++) {}
<297> }
```

```
<298> while (b_isWaiting)
<299> {
<300> MSG    Msg;
<301> //if( g_pRemote->b_isBeforeUpdate && g_b_isSyncStep2) b_isWaiting = FALSE;
<302> if( g_pRemote->b_isBeforeUpdate) b_isWaiting = FALSE;
<303> //for exit this loop
<304> if ( IsKeyDown( VK_ESCAPE ) )
<305> {
<306> //jeEngine_Printf(pM2p->Engine, 100, 100, "have been pressed ESC key");
<307> b_isWaiting = FALSE;
<308> return FALSE;
<309> }
<310> //---------------
<311> if(PeekMessage(&Msg, NULL, 0, 0, PM_REMOVE))    //non-blocking message check
<312> {
<313> // stop after quit message is posted
<314> if (Msg.message == WM_QUIT)
<315> // break;       // <---loop ends here
<316> return FALSE;
<317> TranslateMessage(&Msg);
```

```
<318> DispatchMessage(&Msg);
<319> }
<320> }
<321> myLog_Printf("COMPLETE SYNCRONIZED STEP2 ==========> ₩n");
<322> g_b_isFirstEntryToUpdate = TRUE;
<323> }
<324> //--------------------------
<325> // f_current_time을 이원화 함.
<326> // MP3가 재생되지 않을때는 timeGetTime()에서 데이터를 추출.
<327> // MP3가 재생될 ??는 jeMP3_GetCurrentPosition()을 이용해서 데이터를 추출.
<328> //if ( !pM2p->myMP3->b_isTimerActive)
<329> if ( !pM2p->pMyMP3->b_isMP3Playing)
<330> {
<331> f_current_time = (float) timeGetTime();
<332> f_DeltaTime = (0.001f)*(f_current_time - f_OldTimeGetTime);
<333> f_OldTimeGetTime = f_current_time;
<334> }
<335> else
<336> {
<337> f_DeltaTime = pM2p->pMyMP3->f_delta_time;
```

```
<338> f_current_time = (float) timeGetTime();
<339> f_OldTimeGetTime = f_current_time;
<340> }
<341> //------------------------
<342> // update multiplay two player
<343> if(
<344> !M2P_Update(
<345> pM2p, f_DeltaTime,
<346> g_b_isSyncStep2, g_b_isSyncStep3,
<347> g_pRemote
<348> )
<349> )
<350> {
<351> return FALSE;
<352> }
<353> Preset->Camera = ViewMgr_GetCamera( pM2p->pView );
<354> assert( Preset->Camera != NULL );
<355> //all done
<356> return TRUE;
<357> } // BOOL Update()
```

```
<358> ////////////////////////////////////////////////////////////////////////////////////////////////
      ////////////////////////////////////////////////
<359> jeBoolean M2P_Update(
<360> M2PInfo *pM2p, float f_DeltaTime,
<361> BOOL b_isSync_PrevStep, BOOL b_isSync_CurrentStep,
<362> GAMEMSG_GENERIC2 *pRemote )
<363> {
<364> //-------------
<365> //stage clear signal proccessing
<366> //(myMP3->b_isBeatScoreEnd 처리)
<367> // send a signal from myMP3Mgr to timeline
<368> if(pM2p->pMyMP3->b_isBeatScoreEnd)
<369> {
<370> pM2p->pDancer->TimeLine->b_isStageClear = JE_TRUE;
<371> /*  myLog_Printf(
<372> "pM2p->pDancer->TimeLine->b_isStageClear :%d₩n",
<373> pM2p->pDancer->TimeLine->b_isStageClear
<374> );
<375> */ }
<376> //-------------
```

```
<377> if(!M2P_Update_Play(pM2p, f_DeltaTime, b_isSync_PrevStep, b_isSync_CurrentStep, pRemote
     )) return JE_FALSE;
<378> // ALL DONE
<379> return JE_TRUE;
<380> } // M2P_Update()
<381> //-------------------------------------
<382> jeBoolean M2P_Update_Play(
<383> M2PInfo *pM2p, float f_DeltaTime,
<384> BOOL b_isSync_PrevStep, BOOL b_isSync_CurrentStep,
<385> GAMEMSG_GENERIC2 *pRemote )
<386> {
<387> int i = 0;
<388> // 게임 시작?
<389> if (!pM2p->b_isBeforeUpdate_MP3 )
<390> {
<391> if (!M2P_IsGameStart(pM2p)) return JE_FALSE;
<392> if (pM2p->b_isBeforeUpdate_MP3) pM2p->b_isGameStarted = TRUE;
<393> //-------------------------
<394> // trans game data : befor_update_mp3
<395> if(pM2p->b_isBeforeUpdate_MP3)// && b_isSync_PrevStep)
```

```
<396> {
<397> {
<398> HRESULT hr = S_OK;
<399> hr = SendTo_GameData(GAME_MSGID_BEFORE_UPDATE_MP3, (LONG)pM2p->b_isBeforeUpdate_MP3,
      TRUE);
<400> if( FAILED(hr) )
<401> {
<402> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<403> return JE_FALSE;
<404> }
<405> }
<406> }
<407> }


<408> if(pRemote->b_isBeforeUpdate_MP3 && pM2p->b_isGameStarted)
<409> {
<410> if (g_b_isFirstTime)
<411> {
<412> myLog_Printf("COMPLETE SYNCRONIZED STEP3 ==========> ₩n");
<413> g_b_isFirstTime = FALSE;
```

```
<414> {
<415> HRESULT hr = S_OK;
<416> pM2p->jeb_isStartMP3 = JE_TRUE;
<417> //myLog_Printf("pM2p->jeb_isStartMP3 : %dWn", pM2p->jeb_isStartMP3);
<418> hr = SendTo_GameData(GAME_MSGID_START_MP3, (LONG)pM2p->jeb_isStartMP3, TRUE);
<419> if( FAILED(hr) )
<420> {
<421> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<422> return JE_FALSE;
<423> }
<424> }
<425> }
<426> // game Quit
<427> if(pM2p->jeb_isGameQuit || pRemote->b_isGameQuit)
<428> {
<429> myMP3Mgr_Destroy( &pM2p->pMyMP3,  &pM2p->pTP );
<430> pM2p->pMyMP3 = NULL;
<431> pM2p->pTP = NULL;
<432> //----------
<433> if (!M2P_IsGameQuit(pM2p)) return JE_FALSE;
```

```
<434> }
<435> // 게임오버시..
<436> else if (pM2p->b_isGameOver || pRemote->b_isGameOver)
<437> {
<438> myMP3Mgr_Destroy( &pM2p->pMyMP3,  &pM2p->pTP );
<439> pM2p->pMyMP3 = NULL;
<440> pM2p->pTP = NULL;
<441> //-----------
<442> if (!M2P_IsGameOver(pM2p)) return JE_FALSE;
<443> }
<444> // 스테이지 클리어..
<445> else if ( pM2p->b_isStageClear || pRemote->b_isGameClear )
<446> {
<447> myMP3Mgr_Destroy( &pM2p->pMyMP3,  &pM2p->pTP );
<448> pM2p->pMyMP3 = NULL;
<449> pM2p->pTP = NULL;
<450> //-----------
<451> if (!M2P_IsStageClear(pM2p)) return JE_FALSE;
<452> }
<453> // 게임 진행..
```

```
<454> else if((pM2p->jeb_isStartMP3 && pRemote->b_isStartMP3)||(pRemote->b_isStartMP3 &&
     pM2p->jeb_isStartMP3))
<455> {
<456> //   for ( i = 0; i < 10000; i++){}
<457> // for debug print
<458> // From pM2p->pDancer->b_isprint
<459> // to pM2p->pMyMP3->b_isprint
<460> pM2p->pMyMP3->b_isprint = pM2p->pDancer->b_isprint;
<461> if (pM2p->pMyMP3 !=NULL) myMP3Mgr_Update(pM2p->pMyMP3, pRemote);
<462> // for debug print
<463> // From pM2p->pDancer->b_isprint
<464> // to pM2p->pMyMP3->b_isprint
<465> pM2p->pDancer->b_isprint= pM2p->pMyMP3->b_isprint;
<466> jeWorld_Frame(pM2p->pWorld->World, f_DeltaTime);
<467> //---------------------------
<468> if(!pM2p->b_isGameOver || !pM2p->b_isStageClear || !pM2p->jeb_isGameQuit)
<469> {
<470> Dancer_Update( pM2p->pDancer, pM2p->Engine, f_DeltaTime,
<471> pM2p->pMyMP3->i_currentBeatIndex,
```

```
<472> pRemote );
<473> }
<474> //------------------------
<475> jeVec3d_Copy(&pM2p->pDancer->v_originPointsOfCameraSpin, &pM2p->pView->
      v_originPositionOfSpin);
<476> // check game stat
<477> pM2p->b_isGameOver = Dancer_IsGameOver( pM2p->pDancer );
<478> if(pM2p->b_isGameOver)
<479> {
<480> {
<481> HRESULT hr = S_OK;
<482> hr = SendTo_GameData(GAME_MSGID_GAME_OVER, (LONG)pM2p->b_isGameOver, TRUE);
<483> if( FAILED(hr) )
<484> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<485> }
<486> }
<487> // check game stat
<488> pM2p->b_isStageClear = Dancer_IsStageClear( pM2p->pDancer );
<489> if (pM2p->b_isStageClear)
<490> {
```

```
<491> {
<492> HRESULT hr = S_OK;
<493> hr = SendTo_GameData(GAME_MSGID_GAME_CLEAR, (LONG)pM2p->b_isStageClear, TRUE);
<494> if( FAILED(hr) )
<495> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<496> }
<497> }
<498> }
<499> }
<500> else jeEngine_Printf(pM2p->Engine, 250, 250, "waiting ...");
<501> // all done
<502> return TRUE;
<503> }
<504> ////////////////////////////////////////////////////////////////////////////////////////////////
     ////////////////////////////////////////////////
<505> void Dancer_Update(
<506> DancerInfo  *pD,   // tester to move
<507> jeEngine  *Engine,
<508> float   f_DeltaTime,
<509> int   i_BeatIndex_fromMP3,
```

```
<510> GAMEMSG_GENERIC2 *pRemote
<511>  )
<512> {
<513> // local
<514> float f_Tempo = 0.0f;
<515> // ensure valid data
<516> assert( pD != NULL );
<517> assert( Engine !=NULL);
<518> //----------------------------
<519> f_Tempo = TimeLine_getTempo( pD->TimeLine );
<520> pD->f_Tempo = f_Tempo;
<521> if ( Dancer_IsBeatScoreOK(pD) == JE_FALSE )
<522> {
<523> pD->m_current.i_numOfbar = 1;
<524> //TimeLine_SetMotionIndex( pD->TimeLine, pD->m_current.i_index );
<525> }
<526> else
<527> {
<528> pD->m_current.f_time = pD->m_current.f_time + (f_DeltaTime * f_Tempo );
<529> //---------
```

```
<530> if ( pD->m_current.f_time < pD->m_current.f_lenght )
<531> {
<532> if(pRemote->b_isLeader_local) Dancer_Update_setKeyInput_Leader(pD, Engine, pRemote);
<533> else       Dancer_Update_setKeyInput_Follower(pD, Engine, pRemote);
<534> }
<535> //---------
<536> pD->b_isMotionEnd = JE_FALSE;
<537> //--------------------------------
<538> //motion end process
<539> if (pD->m_current.f_time >= pD->m_current.f_lenght )
<540> {
<541> Dancer_Motion_Ending(pD);
<542> //----------------
<543> pD->b_isStepscore_display = FALSE;
<544> pD->b_isStepscore_full  = FALSE;
<545> Dancer_Update_setMotionEnd_Proc(pD, Engine, pRemote);
<546> } //motion end process
<547> //--------------------------------
<548> // motion render process
<549> if (pD->b_isCurrentMotionProc) Dancer_Motion_Render(pD);
```

```
<550> else       pD->b_isMotionEnd = JE_TRUE;
<551> // motion processing
<552> }
<553> //------------------------
<554> // for debugging
<555> // Dancer_Check_TimeLine( pD, Engine );
<556> //----------------
<557> // rtp sync
<558> {
<559> float  f_term = 0.0f;
<560> BOOL  b_rtpIsChange = FALSE;
<561> pD->i_rtp_percent_prev = pD->i_rtp_percent;
<562> if (pD->m_current.f_time == 0)
<563> {
<564> f_term = 1;
<565> pD->i_rtp_percent = 100;
<566> }
<567> else if ( (pD->m_current.f_time > 0) && (pD->m_current.f_time < pD->
     m_current.f_lenght) )
<568> {
```

```
<569> f_term = 1 - (pD->m_current.f_time/pD->m_current.f_lenght);
<570> pD->i_rtp_percent = (int) ( f_term * 100 );
<571> }
<572> else if ( pD->m_current.f_time >= pD->m_current.f_lenght)
<573> {
<574> f_term = 0;
<575> pD->i_rtp_percent = 0;
<576> }
<577> if (pD->i_rtp_percent_prev == pD->i_rtp_percent) b_rtpIsChange = FALSE;
<578> else b_rtpIsChange = TRUE;
<579> if (!pD->Energy->bIsGameOver) TimeLine_Update(
<580> pD->TimeLine, Engine, i_BeatIndex_fromMP3,
<581> pD->i_rtp_percent, b_rtpIsChange
<582> );
<583> }
<584> //---------------
<585> Energy_Update(pD->Energy, Engine);
<586> } // Dancer_Update()
<587> //----------------------------------------
```

```
<588> void Dancer_Update_setKeyInput_Leader(DancerInfo *pD, jeEngine *Engine, GAMEMSG_GENERIC2
     *pRemote)
<589> {
<590> //---------
<591> // notify key input time to the time line
<592> pD->b_isInputTime = JE_TRUE;
<593> if ( ( pD->b_isDisableKeyInput_fromPlayer == JE_FALSE ) )
<594> pD->b_isOK_KeyInput_fromPlayer = Dancer_MotionIndex_Assignments( pD );
<595> //-------------
<596> if ( pD->b_isOK_KeyInput_fromPlayer )
<597> {
<598> // stepscore display control
<599> // b_isStepscore_display : FALSE: Disable display;
<600> // b_isStepscore_full :  TRUE : fulled image, FALSE : empty image
<601> pD->b_isStepscore_display = TRUE;
<602> pD->b_isStepscore_full  = FALSE;
<603> // game data sending
<604> pD->i_display_index = pD->m_generic.i_index;
<605> if(!pD->b_isTransToRemote)
<606> {
```

```
<607> {
<608> HRESULT hr = S_OK;
<609> hr = SendTo_GameData(GAME_MSGID_MOTION_REQUEST, (LONG)pD->m_generic.i_index, FALSE);
<610> if( FAILED(hr) )
<611> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<612> }
<613> // for debug print
<614> //pD->b_isprint = TRUE;
<615> pD->b_isTransToRemote = JE_TRUE;
<616> }
<617> // Dancer_MotionIndex_Assignments() closing.
<618> pD->b_isDisableKeyInput_fromPlayer = TRUE;
<619> }
<620> if ( pD->b_isOK_KeyInput_fromPlayer )
<621> {
<622> if (pRemote->l_MotionIndex_Request != -1)
<623> {
<624> // is next motion OK, or fail?
<625> if ( pD->m_generic.i_index != pRemote->l_MotionIndex_Request )
<626> {
```

```
<627> pD->b_isKeyInputOK = JE_FALSE;
<628> }
<629> else
<630> {
<631> // stepscore display control
<632> pD->b_isStepscore_display = TRUE;
<633> pD->b_isStepscore_full  = TRUE;
<634> // get Motion data
<635> Dancer_GetMotionData(pD);
<636> //-------------------------
<637> // for prev. input
<638> Dancer_SetNextMotion(pD);
<639> pD->b_isKeyInputOK = JE_TRUE;
<640> }
<641> myLog_Printf("pD->b_isKeyInputOK : %d₩n", pD->b_isKeyInputOK);
<642> //-------------------------
<643> pD->b_isOK_KeyInput_fromPlayer = JE_FALSE;
<644> }
<645> }
<646> //-------------
```

```
<647> } // void Dancer_Update_setKeyInput_Leader()
<648> //-----------------------------------------
<649> void Dancer_Update_setKeyInput_Follower(DancerInfo *pD, jeEngine *Engine,
      GAMEMSG_GENERIC2 *pRemote)
<650> {
<651> //---------
<652> // notify key input time to the time line
<653> pD->b_isInputTime = JE_TRUE;
<654> // recieved motion index from the leader
<655> if (pRemote->l_MotionIndex_Request != -1)
<656> {
<657> // stepscore display control
<658> // b_isStepscore_display : FALSE: Disable display;
<659> // b_isStepscore_full :  TRUE : fulled image, FALSE : empty image
<660> pD->b_isStepscore_display = TRUE;
<661> pD->b_isStepscore_full  = FALSE;
<662> pD->i_display_index = pRemote->l_MotionIndex_Request;
<663> if ( ( pD->b_isDisableKeyInput_fromPlayer == JE_FALSE ) )
<664> pD->b_isOK_KeyInput_fromPlayer = Dancer_MotionIndex_Assignments( pD );
<665> }
```

```
<666> //------------
<667> if (pRemote->l_MotionIndex_Request != -1)
<668> {
<669> if ( pD->b_isOK_KeyInput_fromPlayer )
<670> {
<671> // game data sending
<672> if(!pD->b_isTransToRemote)
<673> {
<674> {
<675> HRESULT hr = S_OK;
<676> hr = SendTo_GameData(GAME_MSGID_MOTION_REQUEST, (LONG)pD->m_generic.i_index, FALSE);
<677> if( FAILED(hr) )
<678> MessageBox( NULL, TEXT("FAILED. :("), TEXT("g_pDP->SendTo"), MB_OK );
<679> }
<680> // for debug print
<681> //pD->b_isprint = TRUE;
<682> pD->b_isTransToRemote = JE_TRUE;
<683> }
<684> // Dancer_MotionIndex_Assignments() closing.
<685> pD->b_isDisableKeyInput_fromPlayer = TRUE;
```

```
<686> // is next motion OK, or fail?
<687> if ( pD->m_generic.i_index != pRemote->l_MotionIndex_Request )
<688> {
<689> pD->b_isKeyInputOK = JE_FALSE;
<690> }
<691> else
<692> {
<693> // stepscore display control
<694> pD->b_isStepscore_display = TRUE;
<695> pD->b_isStepscore_full  = TRUE;
<696> // get Motion data
<697> Dancer_GetMotionData(pD);
<698> //-------------------------
<699> // for prev. input
<700> Dancer_SetNextMotion(pD);
<701> pD->b_isKeyInputOK = JE_TRUE;
<702> }
<703> myLog_Printf("pD->b_isKeyInputOK : %dWn", pD->b_isKeyInputOK);
<704> //-----------------------
<705> pD->b_isOK_KeyInput_fromPlayer = JE_FALSE;
```

```
<706> }
<707> }
<708> if (pD->b_isKeyInputOK)
<709> {
<710> pD->b_isStepscore_display = TRUE;
<711> pD->b_isStepscore_full  = TRUE;
<712> }
<713> //------------
<714> } // void Dancer_Update_setKeyInput_Follower()
<715> //-----------------------------------------
<716> //void Dancer_Update_setMotionEnd_Proc(DancerInfo *pD)//, float f_Tempo)
<717> void Dancer_Update_setMotionEnd_Proc(DancerInfo *pD, jeEngine *Engine, GAMEMSG_GENERIC2
     *pRemote)
<718> {
<719> {
<720> if (pRemote->l_MotionIndex_Request == -1) pD->b_isKeyInputFailed = JE_TRUE;
<721> if (!pD->b_isKeyInputOK) pD->b_isKeyInputFailed   = JE_TRUE;
<722> else    pD->b_isKeyInputFailed  = JE_FALSE;
<723> pD->b_isInputTime = JE_FALSE;
<724> pD->b_isTransToRemote = JE_FALSE;
```

```
<725> // remote motion index init.
<726> pRemote->l_MotionIndex_Request = -1;
<727> }
<728> if (pD->b_isKeyInputFailed)
<729> {
<730> Dancer_NextMotionKeyInputISFail(pD, pD->TimeLine, Engine );
<731> }
<732> //------------------
<733> // Health Adjustment
<734> Dancer_HealthStats_Adjustments( pD, Engine );
<735> //---------
<736> //----------------
<737> Dancer_SetCurrentMotion(pD);
<738> Dancer_GetBarsOfMotion( pD);//, f_Tempo );
<739> pD->b_isMotionEnd = JE_FALSE;
<740> pD->m_current.f_time = 0.0f;
<741> pD->b_isMotionEnd = JE_TRUE;
<742> pD->b_isKeyInputOK = JE_FALSE;
<743> pD->b_isDisableKeyInput_fromPlayer = JE_FALSE;// why ture?
<744> pD->b_isEnergyChange = JE_TRUE;
```

```
<745> }
<746> ////////////////////////////////////////////////////////////////////////////////////////////////
      ////////////////////////////////////////////////
<747> myMP3MgrInfo * myMP3Mgr_Create(
<748> HWND hWnd,
<749> jeEngine *Engine,
<750> jeWorld *World,
<751> BeatScorePool *BeatPool, // Beat Score Pool
<752> const char* DebugPath
<753> )
<754> {
<755> // local variable
<756> myMP3MgrInfo *myMP3;
<757> TimerProcInfo *TP;
<758> // ensure valid data
<759> // create MP3 manager structue
<760> myMP3 = jeRam_Allocate( sizeof( myMP3MgrInfo ) );
<761> if ( myMP3 == NULL ) return NULL;
<762> mymemset( myMP3, 0, sizeof( myMP3MgrInfo ) );
<763> // create TimerProcInfo structue
```

```
<764> TP = jeRam_Allocate( sizeof( TimerProcInfo ) );
<765> if ( TP == NULL ) return NULL;
<766> mymemset( TP, 0, sizeof( TimerProcInfo ) );
<767> // structer Init.
<768> myMP3Mgr_Init(myMP3);
<769> strcpy(Debug_Path ,DebugPath);
<770> //------------------------
<771> myMP3->Engine = Engine;
<772> myMP3->hWnd  = hWnd;
<773> myMP3->World = World;
<774> // create sound system
<775> myMP3Mgr_CreateSoundSystem(myMP3);
<776> // loading MP3
<777> myMP3Mgr_LoadingMP3(myMP3);
<778> //------------------------
<779> // first MP3 play
<780> if (!myMP3->b_isMP3Playing)
<781> {
<782> myMP3->LogoFinish = jeMp3_StandbyPlay(myMP3->SoundSystem, DM1);
<783> }
```

```
<784> //------------------------------------
<785> myMP3->BeatPool = BeatPool;
<786> //------------------------------------------------
<787> myMP3->i_currentBeatIndex = 0;
<788> myMP3->f_currntBeatTimeStamp = BeatScorePool_Get( myMP3->BeatPool, myMP3->
      i_currentBeatIndex );
<789> myMP3->f_PrevPlayTime = BeatScorePool_GetPlayTime( myMP3->BeatPool, myMP3->
      i_currentBeatIndex );
<790> // for stage Clear test
<791> //myMP3->i_beatUpperLimit = 6 +1;
<792> //myMP3->i_beatUpperLimit = myMP3->BeatPool->BeatScoreCount +1;
<793> myMP3->i_beatUpperLimit = myMP3->BeatPool->BeatScoreCount -1;
<794> myMP3Mgr_TimerSetResolution( myMP3 );
<795> //------------------------------------
<796> TP->BeatPool = BeatPool;
<797> TP->i_beatUpperLimit = myMP3->i_beatUpperLimit;
<798> TP->i_currentBeatIndex = 0;
<799> TP->f_PrevPlayTime = myMP3->f_PrevPlayTime;
<800> TP->b_isBeatScoreEnd = myMP3->b_isBeatScoreEnd;
<801> //------------------------------------
```

```
<802> S_myMP3 = TP;
<803> //------------------------------------
<804> // all done
<805> return myMP3;
<806> }
<807> //------------------------------------------------------------------------------//
<808> // myMP3Mgr_Update()
<809> //
<810> // Main application function.
<811> //------------------------------------------------------------------------------
<812> jeBoolean myMP3Mgr_Update(myMP3MgrInfo *myMP3, GAMEMSG_GENERIC2 *pRemote)
<813> {
<814> //ensure valid data
<815> assert(myMP3);
<816> myMP3Mgr_transData_myMP3Mgr2STP( myMP3);
<817> //--------------------
<818> /*  if (myMP3->LogoFinish == 0 && pRemote->b_isMP3Started == TRUE )
<819> */  {
<820> jeMp3_Play(myMP3->SoundSystem, 0, JE_TRUE);
<821> myMP3->b_isMP3Playing = JE_TRUE;
```

```
<822> //--------------------
<823> // for myMP3Mgr_TimerProc()
<824> // f_DelayTime : unit : milli-sec
<825> if ( myMP3Mgr_TimerSet( myMP3, 1.0f ) ) myMP3->b_isTimerActive = JE_TRUE;
<826> else  myMP3->b_isTimerActive = JE_FALSE;
<827> //--------------------
<828> }
<829> /*  else if (myMP3->LogoFinish == 0)
<830> {
<831> jeMp3_Play(myMP3->SoundSystem, 0, JE_TRUE);
<832> myMP3->b_isMP3Playing = JE_TRUE;
<833> jeMp3_Pause(myMP3->SoundSystem);
<834> if (myMP3->b_isMP3Playing)
<835> PostMessage( myMP3->hWnd,
<836> WM_APP_MP3_STATS, GAME_MSGID_MP3_STATS,
<837> myMP3->b_isMP3Playing
<838> );
<839> }
<840> */
<841> // for debugging
```

```
<842> myMP3Mgr_Print_Mp3Property(myMP3);
<843> //-----------------
<844> // for mp3 delta time
<845> myMP3->f_current_time = (float)jeMp3_GetCurrentPosition();
<846> myMP3->f_delta_time = myMP3->f_current_time - myMP3->f_current_time_old;
<847> myMP3->f_current_time_old = myMP3->f_current_time;
<848> myMP3Mgr_transData_STP2myMP3Mgr( myMP3);
<849> /*
<850> myLog_Printf( "S_myMP3->b_isBeatScoreEnd :%d₩n", S_myMP3->b_isBeatScoreEnd );
<851> myLog_Printf( "S_myMP3->i_beatUpperLimit :%d₩n", S_myMP3->i_beatUpperLimit );
<852> myLog_Printf( "S_myMP3->i_currentBeatIndex :%d₩n", S_myMP3->i_currentBeatIndex );
<853> myLog_Printf( "myMP3->b_isBeatScoreEnd :%d₩n", myMP3->b_isBeatScoreEnd );
<854> myLog_Printf( "myMP3->i_beatUpperLimit :%d₩n", myMP3->i_beatUpperLimit );
<855> myLog_Printf( "myMP3->i_currentBeatIndex :%d₩n", myMP3->i_currentBeatIndex );
<856> myLog_Printf( "==========₩n₩n" );
<857> */
<858> return JE_TRUE;
<859> } // myMP3Mgr_Update()
<860> void myMP3Mgr_transData_myMP3Mgr2STP( myMP3MgrInfo *myMP3)
```

```
<861> {
<862> //ensure valid data
<863> assert(myMP3);
<864> S_myMP3->BeatPool = myMP3->BeatPool;
<865> S_myMP3->i_beatUpperLimit = myMP3->i_beatUpperLimit;
<866> S_myMP3->i_currentBeatIndex = myMP3->i_currentBeatIndex;
<867> S_myMP3->f_PrevPlayTime = myMP3->f_PrevPlayTime;
<868> S_myMP3->b_isBeatScoreEnd = myMP3->b_isBeatScoreEnd;
<869> }
<870> void myMP3Mgr_transData_STP2myMP3Mgr( myMP3MgrInfo *myMP3)
<871> {
<872> //ensure valid data
<873> assert(myMP3);
<874> myMP3->BeatPool = S_myMP3->BeatPool;
<875> myMP3->i_beatUpperLimit = S_myMP3->i_beatUpperLimit;
<876> myMP3->i_currentBeatIndex = S_myMP3->i_currentBeatIndex;
<877> myMP3->f_PrevPlayTime = S_myMP3->f_PrevPlayTime;
<878> myMP3->b_isBeatScoreEnd = S_myMP3->b_isBeatScoreEnd;
<879> }
```

【발명의 효과】

<880> 이상에서 상세하게 설명한 이 발명에 따르면, 3차원으로 구현되는 컴퓨터 그래픽스의 영상에서 구조체의 자유로운 동작 표현뿐만 아니라 복수의 링크를 갖는 구조체 간의 상호간섭을 고려한 구조체의 동작을 보다 단순하고 평이한 조작으로 구현할 수 있다.

<881> 또한, 이 발명에 따르면, 3차원 그래픽스 영상을 통하여 제공되는 구조체 동작에서 실제와 유사한 물리적 특성을 구현할 수 있으며, 구조체 전체가 자연스러운 자세를 유지하면서도 연속동작이 가능하게 된다.

<882> 아울러, 이 발명은 통신 가능한 네트워크 상에서 게임자간의 상대적인 관계 및 진행에 의하여 이야기 전개가 변화되는 게임이나 동영상에 활용할 수 있는 효과가 있다.

【특허청구범위】

【청구항 1】

사용자의 요구에 따라 이벤트를 발생시키는 입력부와,

상기 입력부에서 발생한 이벤트를 표준시간과 비교하여 시간차를 보정하도록 하는 동기화부와,

상기 입력부에서 발생한 이벤트가 상기 표준시간에 동기화되도록 연산 및 시스템 제어하는 연산처리부와,

상기 표준시간에 동기된 이벤트에 따른 영상과 음원이 출력되도록 정합하는 인터페이스부와,

상기 인터페이스부를 통해 출력되는 영상과 음원을 출력하는 출력부를 구비한 것을 특징으로 하는 동기화 장치.

【청구항 2】

제 1 항에 있어서, 상기 입력부는 키보드, 마우스, 트랙 볼, 조이스틱, 터치스크린 및 핸드폰 키패드 중 하나 또는 다수의 조합인 것을 특징으로 하는 동기화 장치.

【청구항 3】

제 1 항에 있어서, 상기 입력부는 카메라나 센서를 통한 직접행위입력장치 또는 마이크와 같은 음성입력장치인 것을 특징으로 하는 동기화 장치.

【청구항 4】

제 1 항에 있어서, 상기 동기화부는 상기 표준시간을 설정하는 시간설정부와,

상기 시간설정부에서 설정된 표준시간과 이벤트 전송시간을 계산하여 정보 전송에 따른 이벤트의 시간차를 제거하는 시간동기부와,

구조체를 구성하는 관절 및 이에 의한 이벤트의 동작에 의한 단위동작을 구분하여 진행, 회전, 평행유지 등의 동작을 진행할 수 있도록 하는 단위동작설정부와,

사용자간에 원격으로 접속되어 있는 하나의 구조체와 다른 구조체와의 상호간섭을 고려하기 위해, 설정된 표준시간을 바탕으로 각 사용자에게 동기화된 시간을 토대로 구조체간의 이벤트 및 동작발생의 동시성을 구현하는 단위동작동기부를 구비한 것을 특징으로 하는 동기화 장치.

【청구항 5】

제 4 항에 있어서, 상기 시간설정부는 월드타임코드(WTC)를 상기 표준시간으로 설정하는 것을 특징으로 하는 동기화 장치.

【청구항 6】

제 4 항에 있어서, 상기 단위동작설정부에서의 데이터 생성방식은 각각의 동작에 의한 장면에 해당되는 프레임 정점위치와 데이터를 저장하고 있다가 보간법에 의하여 계산하는 것을 특징으로 하는 동기화 장치.

【청구항 7】

제 4 항에 있어서, 상기 단위동작설정부에서의 데이터 생성방식은 구조체를 여러 개로 나누어서 각각의 관계를 규정하고, 나누어진 구조체의 데이터를 매 프레임 또는 변화되는 프레임마다 지정하여 사용하는 것을 특징으로 하는 동기화 장치.

【청구항 8】

제 4 항에 있어서, 상기 단위동작설정부에서의 데이터 생성방식은 골격이라는 관절단위의 구조체 데이터를 토대로 각각의 관계를 규정한 계층적 구조를 가지고 위치값에 따라 움직이는 것을 특징으로 하는 동기화 장치.

【청구항 9】

제 4 항에 있어서, 상기 단위동작동기부는 단위동작을 맞추기 위해서 부수적으로 음원을 사용하는 것을 특징으로 하는 동기화 장치.

【청구항 10】

제 9 항에 있어서, 상기 음원은 WAV, MP3, WMA, MIDI 중 하나인 것을 특징으로 하는 동기화 장치.

【청구항 11】

제 9 항에 있어서, 상기 단위동작동기부는 음원의 진행시간에 맞추어 표준시간을 설정하고 상기 표준시간을 단위동작과 동기시키는 것을 특징으로 하는 동기화 장치.

【청구항 12】

제 1 항에 있어서, 상기 출력부는 화상출력기와 음원출력기를 구비한 것을 특징으로 하는 동기화 장치.

【청구항 13】

제 12 항에 있어서, 상기 화상출력기는 모니터, HUD(Head Up Display)장치, LCD 패널인 것을 특징으로 하는 동기화 장치.

【청구항 14】

제 12 항에 있어서 상기 음원출력기는 스피커인 것을 특징으로 하는 동기화 장치.

【청구항 15】

제 1 항에 있어서, 상기 화상출력기는 입체로 구성된 대상물과의 송수신하여 상기 입체 대상물을 통해 입출력 매개상태를 확인하는 것을 특징으로 하는 동기화 장치.

【청구항 16】

각종 온라인 및 웹상에서 다수의 클라이언트 시스템이 접속되어 구현되는 가상공간 상에서, 상기 다수의 클라이언트 시스템들을 표준시간으로 동기화하고,

상기 다수의 클라이언트 시스템들 각각에서 발생한 이벤트들을 상대방 클라이언트들에게 전송하고,

상기 일정한 기준구간 동안에 발생한 이벤트들을 상기 표준시간에 동기시켜 화면에 표시하는 것을 특징으로 하는 동기화 방법.

【청구항 17】

제 16 항에 있어서, 상기 가상공간의 시간동기코드를 상기 각 클라이언트 시스템에 동기화한 후 상기 클라이언트 시스템으로부터 입력된 단위동작을 단위동작입력시간 종료후에 시간설정코드에 맞추어 동시에 실행하는 것을 특징으로 하는 동기화 방법.

【청구항 18】

제 16 항에 있어서, 상기 시간동기코드는 월드타임코드인 것을 특징으로 하는 동기화 방법.

【청구항 19】

제 16 항에 있어서, 상기 구조체는 상기 클라이언트 시스템으로부터 입력되는 2이상의 이벤트에 따른 상호간섭에 의하여 하나의 형태로 진행되는 것을 특징으로 하는 동기화 방법.

【청구항 20】

제 16 항에 있어서, 상기 동기화방법은 하나의 서버시스템과 다수의 클라이언트 시스템에 의한 서버/클라이언트 방식으로 운영되는 것을 특징으로 하는 동기화 방법.

【청구항 21】

제 16 항에 있어서, 상기 동기화방법은 다수의 클라이언트 시스템에 의한 피어투피어 방식으로 운영되는 것을 특징으로 하는 동기화 방법.

【청구항 22】

제 21 항에 있어서, 상기 피어투피어 방식은 정보공유형과 자원공유형 중 하나 또는 둘의 결합을 통해 서비스되는 것을 특징으로 하는 동기화 방법.

【청구항 23】

제 21 항에 있어서, 상기 피어투피어 방식은 핑(Ping), 퐁(Pong), 쿼리(Query), 쿼리히트(Queryhit), 푸쉬(Push) 등과 같은 스트립터 중 하나 또는 다수를 사용하는 것을 특징으로

하는 동기화 방법.

【청구항 24】

제 21 항에 있어서, 상기 클라이언트 시스템은 별도의 메모리를 가지고 온라인 혹은 2인 이상 게임이 가능한 PS2 나 X-box, GameCube와 같은 비디오 게임기를 포함하는 것을 특징으로 하는 동기화 방법.

【청구항 25】

각종 온라인 및 웹상에서 표준시간으로 동기화되고,

일정한 기준구간 동안에 발생한 이벤트들을 상호간에 전달하며 상기 표준시간에 동기시켜 구조체를 움직이는 다수의 클라이언트 시스템을 포함한 것을 특징으로 하는 상호작용 시스템.

【청구항 26】

제 25 항에 있어서, 상기 클라이언트 시스템은 사용자의 요구에 따라 이벤트를 발생시키는 입력부와, 상기 입력부에서 발생한 이벤트를 표준시간과 비교하여 시간차를 보정하도록 하는 동기화부와, 상기 입력부에서 발생한 이벤트가 상기 표준시간에 동기화되도록 연산 및 시스템 제어하는 연산처리부와, 상기 표준시간에 동기된 이벤트에 따른 영상과 음원이 출력되도록 정합하는 인터페이스부와, 상기 인터페이스부를 통해 출력되는 영상과 음원을 출력하는 출력부

를 구비한 것을 특징으로 하는 상호작용 시스템.

【청구항 27】

제 26 항에 있어서, 상기 동기화부는 DB처리기와, 가상공간처리기와 개인정보처리기를 구비한 것을 특징으로 하는 상호작용 시스템.

【청구항 28】

제 26 항에 있어서, 상기 연산처리부는 이벤트처리기와 액세스데이터처리기와 GUI처리기를 구비한 것을 특징으로 하는 상호작용 시스템.

【청구항 29】

제 26 항에 있어서, 상기 인터페이스부는 동작진행처리기를 구비한 것을 특징으로 하는 상호작용 시스템.

【청구항 30】

각종 온라인 및 웹상에서 다수의 클라이언트 시스템이 접속되어 구현되는 가상공간 상에서, 상기 다수의 클라이언트 시스템들을 표준시간으로 동기화하고,

상기 다수의 클라이언트 시스템들 각각에서 발생한 이벤트들을 상대방 클라이언트들에게 전송하고,

상기 일정한 기준구간 동안에 발생한 이벤트들에 따른 단위동작을 상기 표준시간에 동기시켜 화면에 표시하여 댄스 게임을 서비스하는 게임방법.

【청구항 31】

제 30 항에 있어서, 상기 단위동작은 처음과 끝의 포즈가 일치하는 것을 특징으로 하는 게임방법.

【청구항 32】

제 30 항에 있어서, 상기 단위동작의 진행시간은 빠르기에 의해 조절되는 것을 특징으로 하는 게임방법.

【청구항 33】

제 30 항에 있어서, 상기 단위동작은 전, 후, 좌, 우, 및 전-좌, 전-우, 후-좌, 후-우측의 8방향으로의 이동을 포함하는 것을 특징으로 하는 게임방법.

【청구항 34】

제 33 항에 있어서, 상기 단위동작은 90도 회전, 180도 회전, 360도 회전 및 특수 단위동작을 포함하는 것을 특징으로 하는 게임방법.

【청구항 35】

제 33 항에 있어서, 상기 단위동작은 앉기, 서기, 구부리기, 연속회전을 포함하는 것을 특징으로 하는 게임방법.

【청구항 36】

제 33 항에 있어서, 상기 단위동작은 구조체를 구성하고 있는 관절 및 이에 의한 동작변형을 포함하는 것을 특징으로 하는 게임방법.

【청구항 37】

제 33 항에 있어서, 상기 단위동작은 구조체를 구성하고 있는 관절 및 이에 의한 다수의 동작의 결합을 포함하는 여러 개를 하나의 단위로 하는 것을 특징으로 하는 게임방법.

【청구항 38】

제 30 항에 있어서, 상기 이벤트는 키보드, 마우스, 조이스틱, 키패널 중 적어도 하나를 이용하여 입력하는 것을 특징으로 하는 게임방법.

【청구항 39】

제 30 항에 있어서, 상기 이벤트는 각종 센서 또는 카메라를 통해 위치값을 입력하여 동작데이터를 입력하는 것을 특징으로 하는 게임방법.

【청구항 40】

제 36 항 또는 제 37 항에 있어서, 상기 구조체의 동작에 대한 조종시, 조종기기의 기계적 제어에 의한 시간적 효과나 항력과 작용/반작용과 같은 공간적, 물리적 효과를 포함하여 처리하는 것을 특징으로 하는 게임방법.

【청구항 41】

제 30 항에 있어서, 상기 구조체는 2차원 또는 3차원 오브젝트인 것을 특징으로 하는 게임방법.

【청구항 42】

제 30 항에 있어서, 상기 구조체는 별도의 모델도구를 통해 만들어진 아바타인 것을 특징으로 하는 게임방법.

【청구항 43】

제 30 항에 있어서, 상기 클라이언트 시스템은 별도의 채팅툴을 포함하여, 상대방 클라이언트 시스템과 문자 혹은 음성으로 의사를 교환하는 것을 특징으로 하는 게임방법.

【청구항 44】

제 41 항에 있어서, 상기 오브젝트는 카메라 등을 통해 입력되는 영상을 바탕으로 제작된 오브젝트와 실제 영상과의 결합을 통해 구현되는 것을 특징으로 하는 게임방법.

【청구항 45】

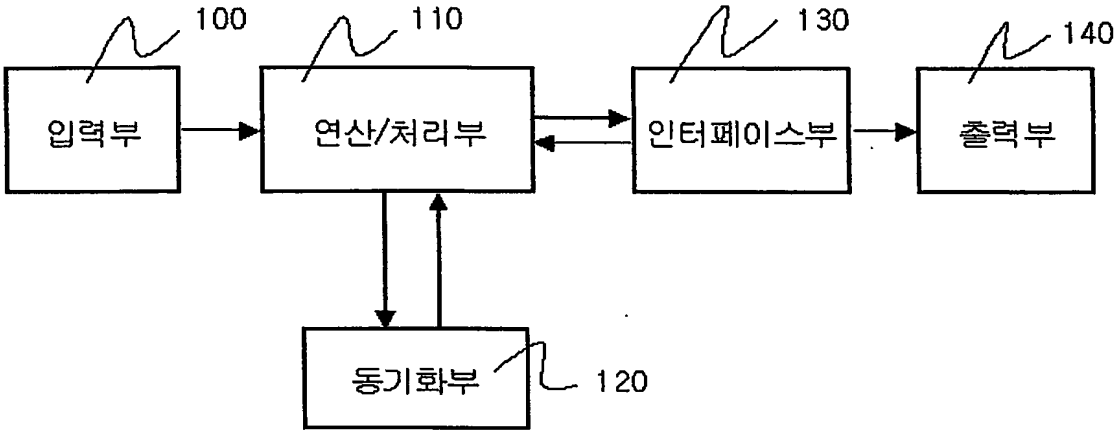
제 30 항에 있어서, 상기 단위동작은 스포츠댄스와 같이 2인이 함께 붙어서 진행되는 것을 특징으로 하는 게임방법.

【청구항 46】

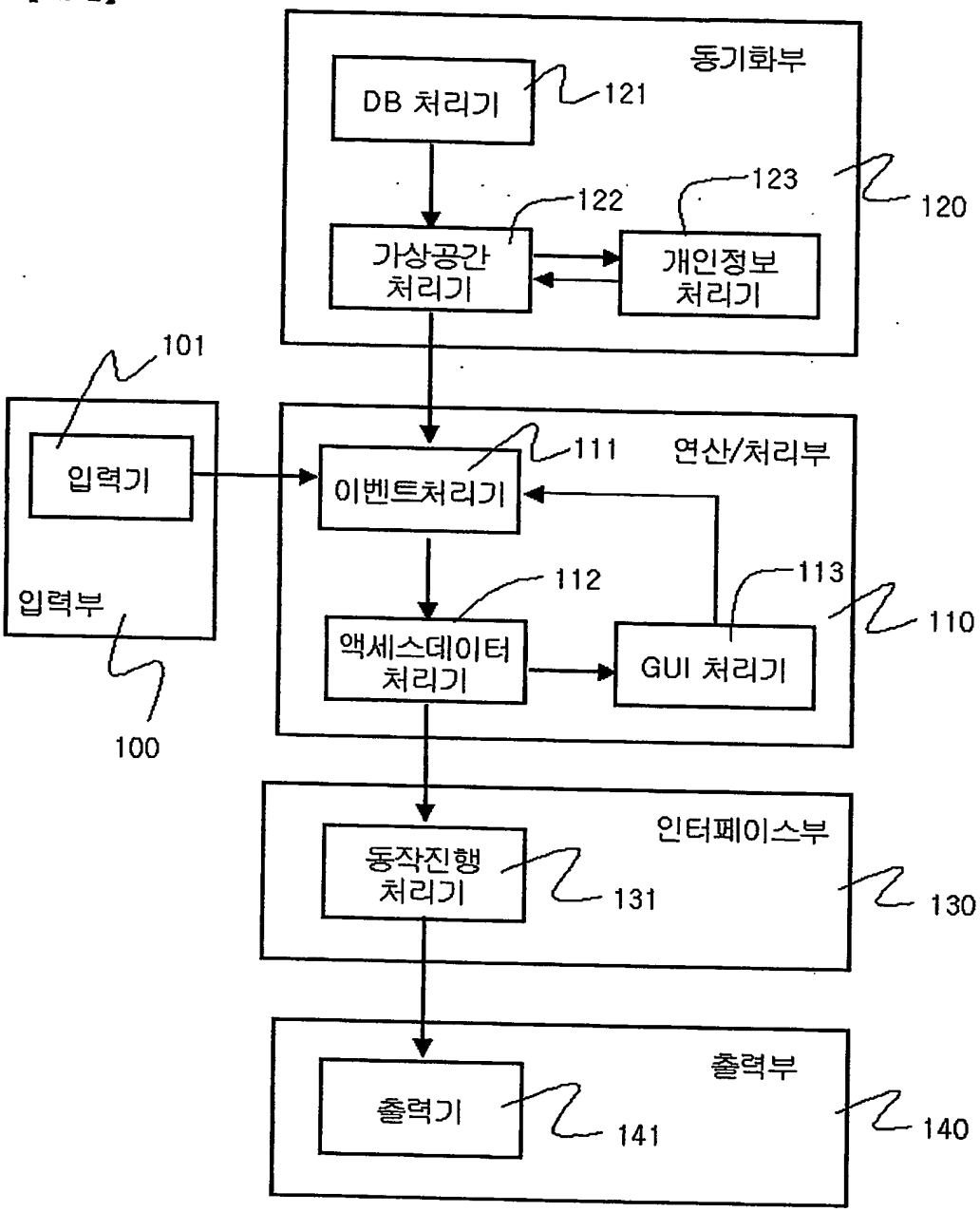
제 45 항에 있어서 스포츠 댄스는 왈츠, 탱고, 폭스트로트, 비엔나 왈츠, 퀵스텝, 자이브, 룸바, 차차차, 삼바, 파도소블, 블루스 중 하나 또는 하나 이상의 결합에 의한 것을 특징으로 하는 게임방법.

【도면】

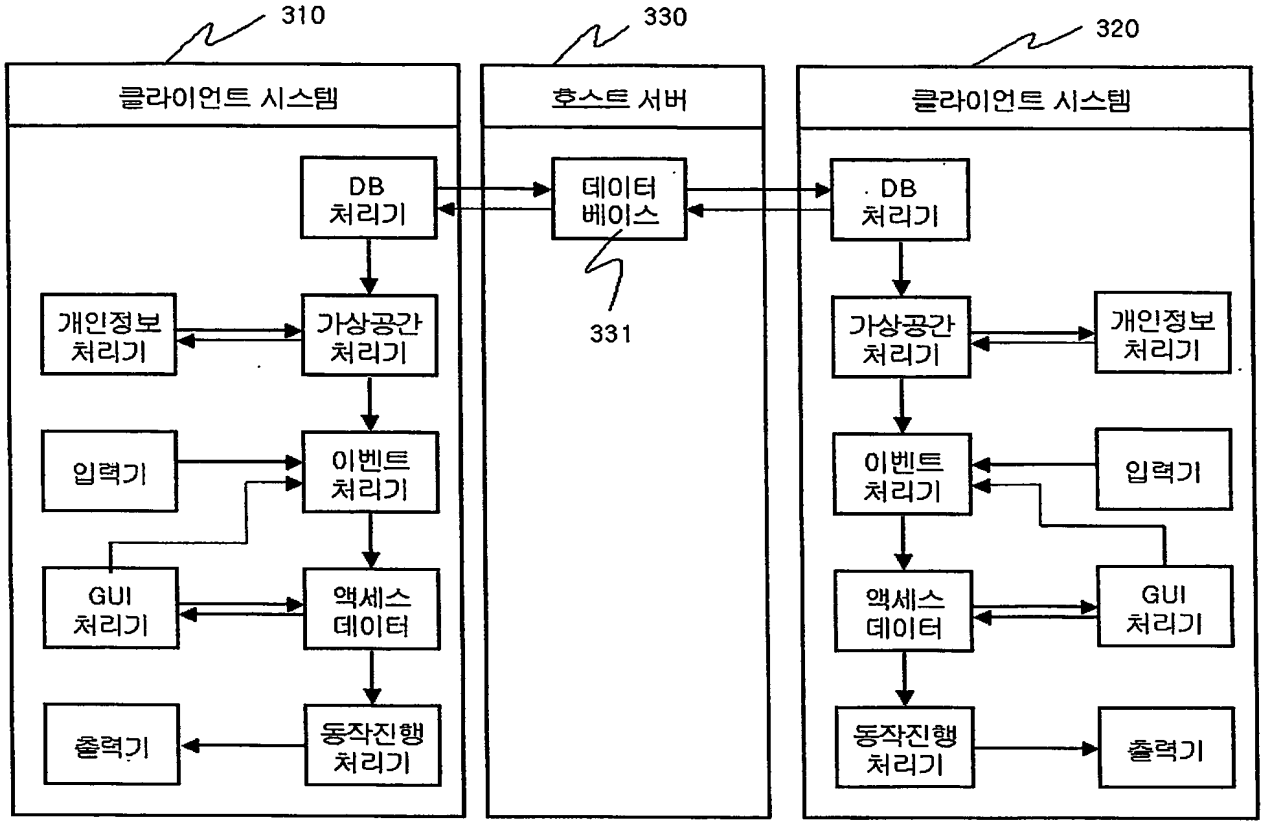
【도 1】



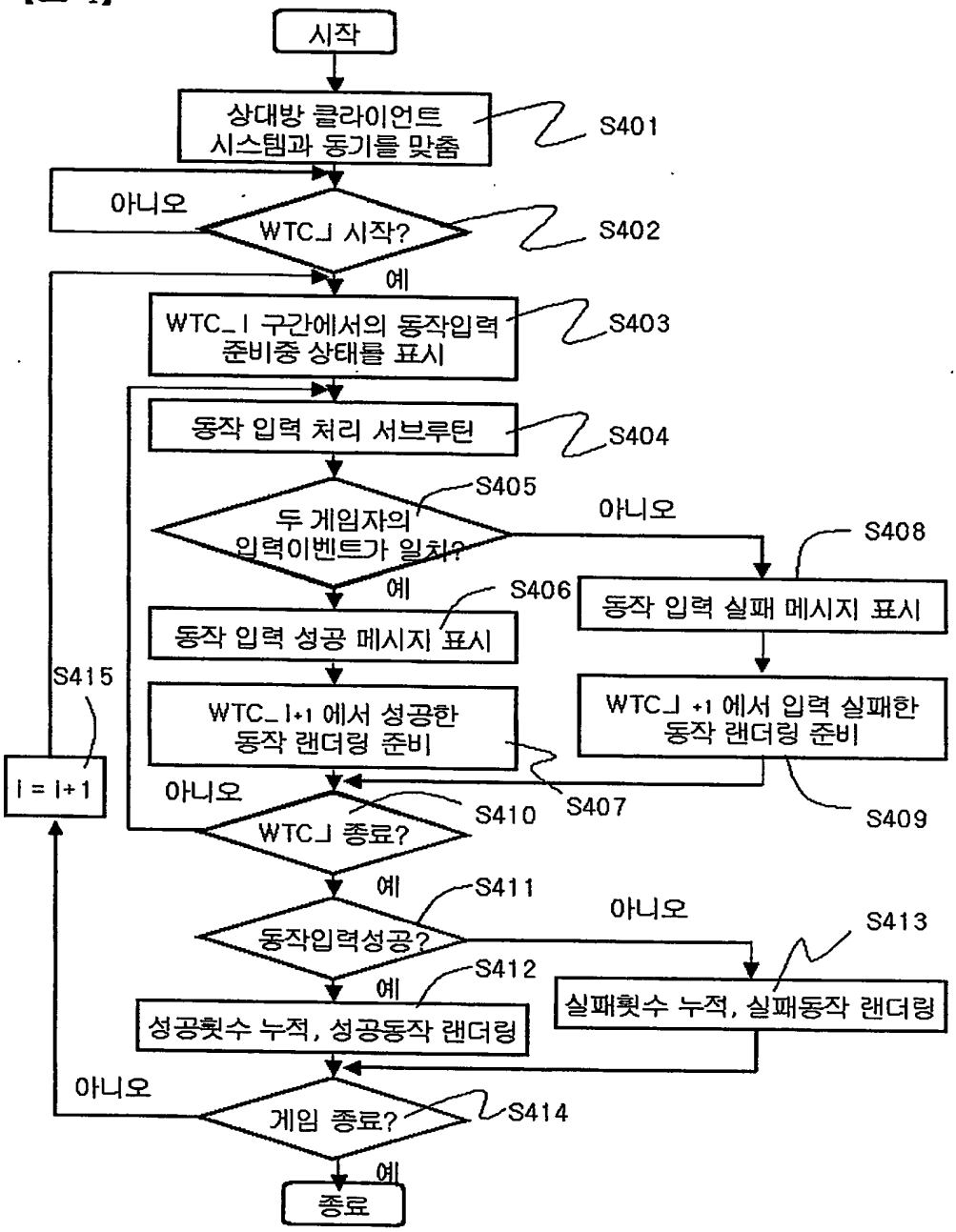
【도 2】


동기화부
DB 처리기
121
123
122
120
가상공간 처리기
개인정보 처리기
101
입력기
111
연산/처리부
이벤트처리기
112
113
입력부
액세스데이터 처리기
GUI 처리기
110
100
인터페이스부
동작진행 처리기
131
130
출력부
출력기
141
140

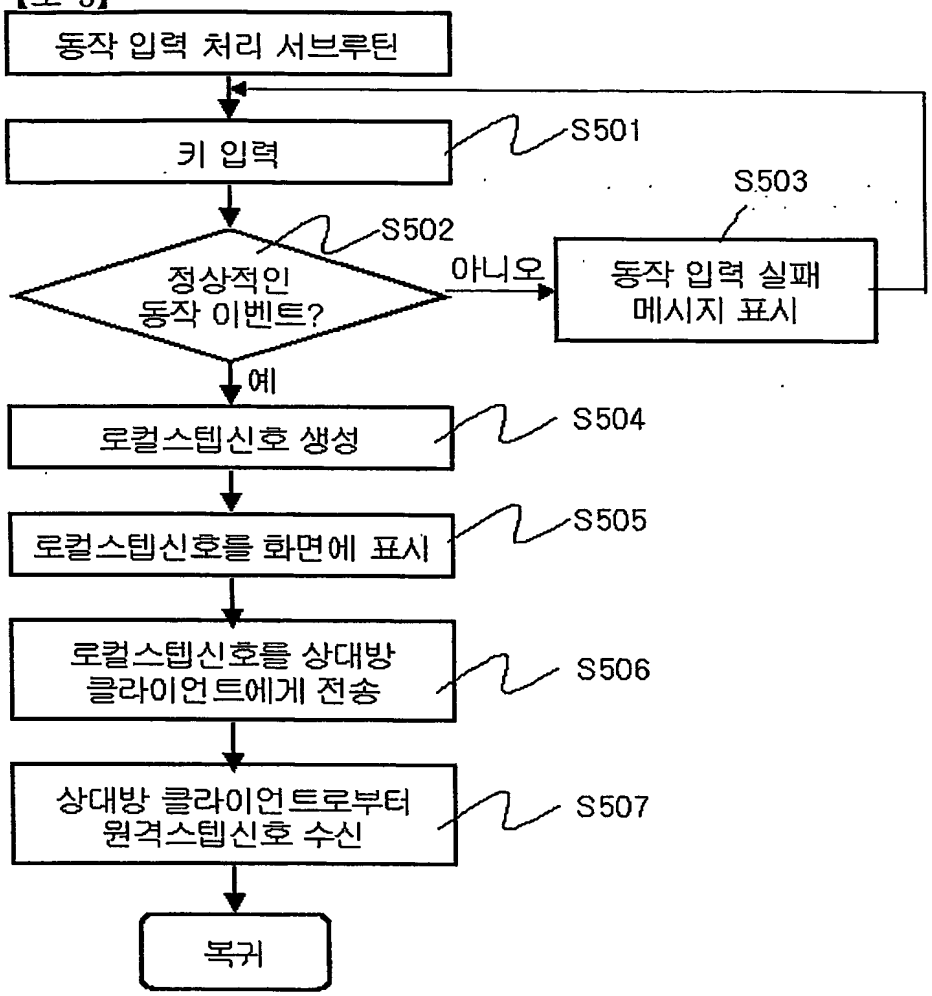
【도 3】


310
330
320
클라이언트 시스템
호스트 서버
클라이언트 시스템
DB 처리기
데이터 베이스
DB 처리기
331
개인정보 처리기
가상공간 처리기
가상공간 처리기
개인정보 처리기
입력기
이벤트 처리기
이벤트 처리기
입력기
GUI 처리기
액세스 데이터
액세스 데이터
GUI 처리기
출력기
동작진행 처리기
동작진행 처리기
출력기

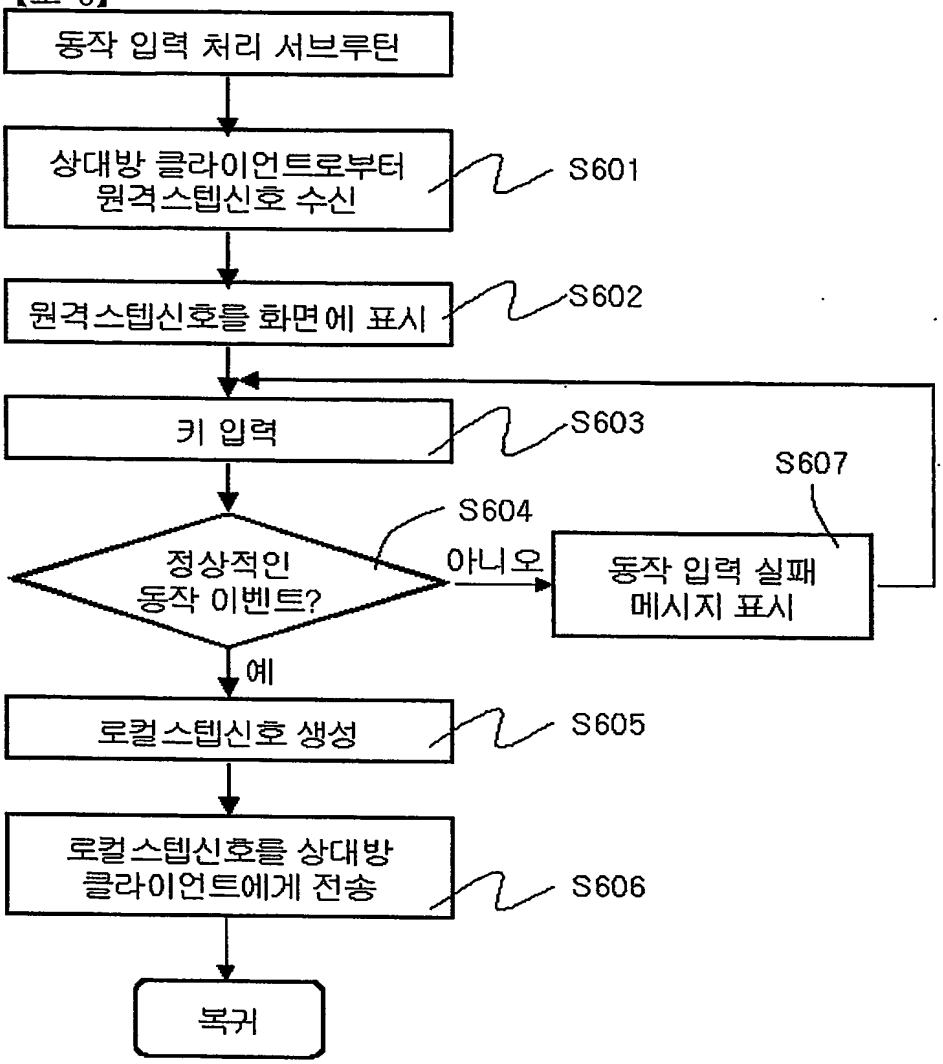
【도 4】


시작
상대방 클라이언트 시스템과 동기를 맞춤
S401
아니오
WTC_I 시작?
S402
예
WTC_I 구간에서의 동작입력 준비중 상태를 표시
S403
동작 입력 처리 서브루틴
S404
S405
두 게임자의 입력이벤트가 일치?
아니오
S408
예
S406
동작 입력 실패 메시지 표시
동작 입력 성공 메시지 표시
S415
WTC_I+1 에서 성공한 동작 랜더링 준비
WTC_I +1 에서 입력 실패한 동작 랜더링 준비
I = I+1
아니오
S410
S407
S409
WTC_I 종료?
예
S411
동작입력성공?
아니오
S413
예
S412
성공횟수 누적, 성공동작 랜더링
실패횟수 누적, 실패동작 랜더링
아니오
게임 종료?
S414
예
종료

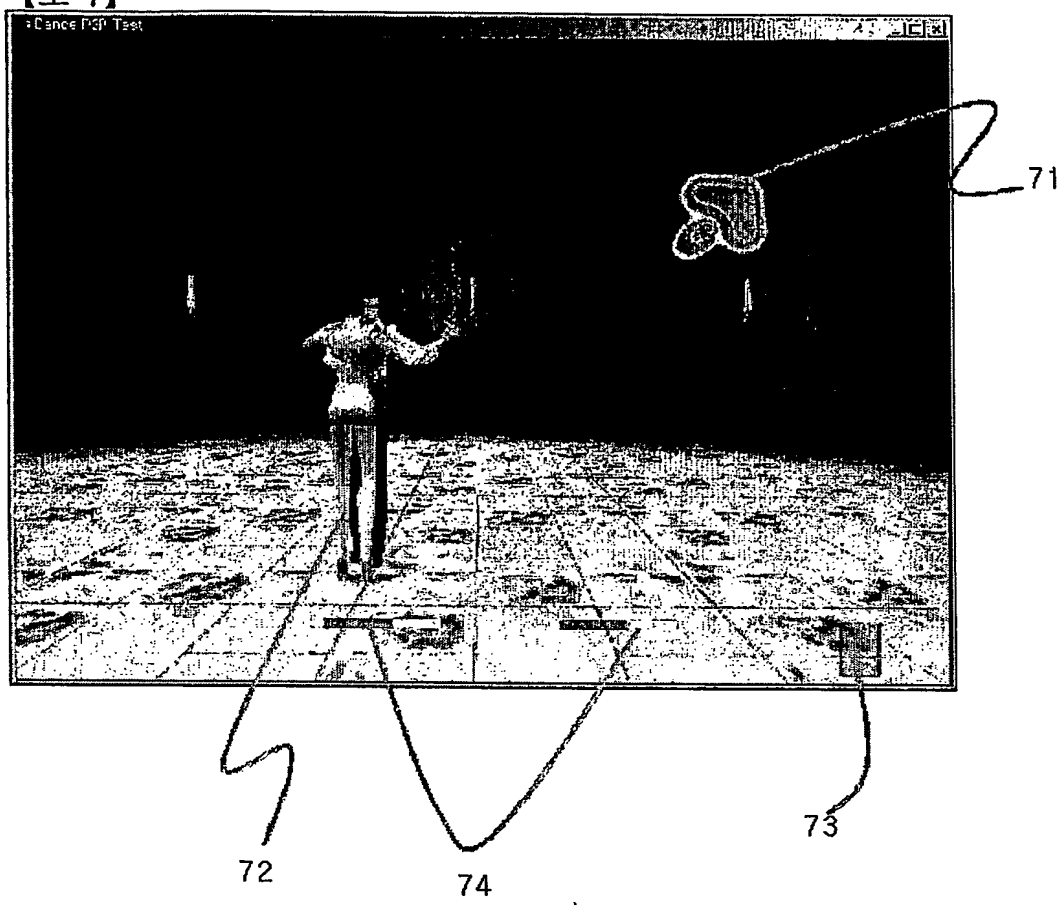
【도 5】


동작 입력 처리 서브루틴
키 입력
S501
정상적인 동작 이벤트?
S502
아니오
동작 입력 실패 메시지 표시
S503
예
로컬스텝신호 생성
S504
로컬스텝신호를 화면에 표시
S505
로컬스텝신호를 상대방 클라이언트에게 전송
S506
상대방 클라이언트로부터 원격스텝신호 수신
S507
복귀

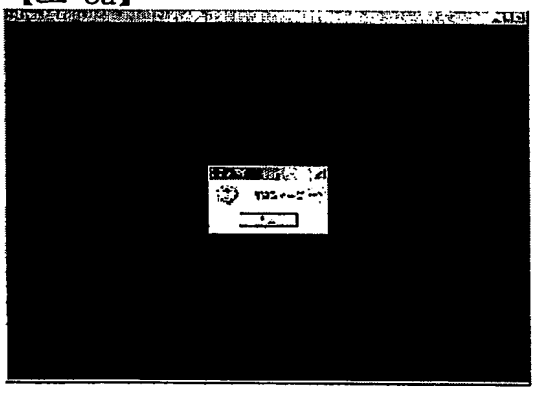
【도 6】


동작 입력 처리 서브루틴
상대방 클라이언트로부터 원격스텝신호 수신
S601
원격스텝신호를 화면에 표시
S602
키 입력
S603
S607
S604
정상적인 동작 이벤트?
아니오
동작 입력 실패 메시지 표시
예
로컬스텝신호 생성
S605
로컬스텝신호를 상대방 클라이언트에게 전송
S606
복귀

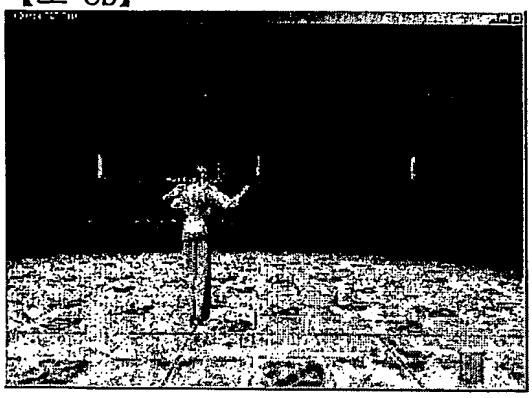
【도 7】



【도 8a】

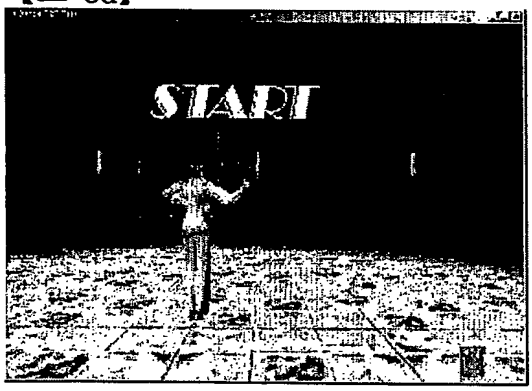


【도 8b】

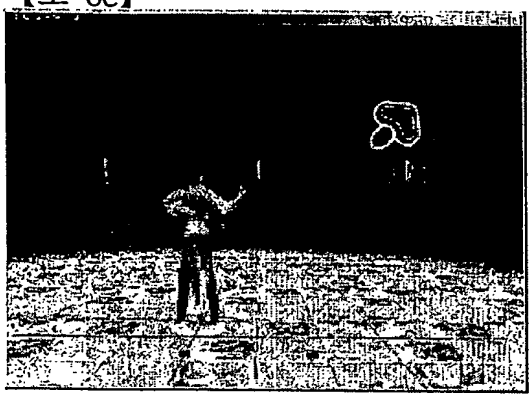


【도 8c】

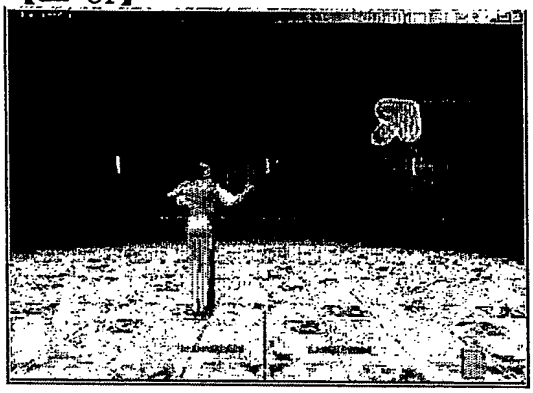
【도 8d】



【도 8e】



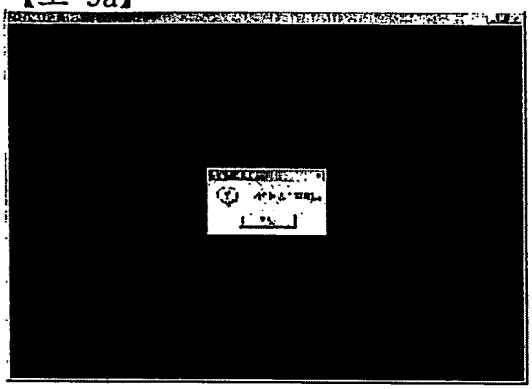
【도 8f】



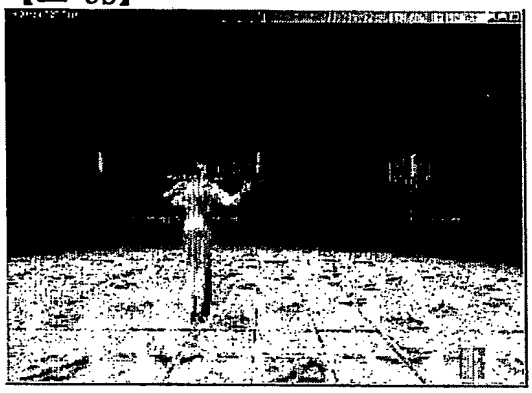
【도 8g】

【도 8h】

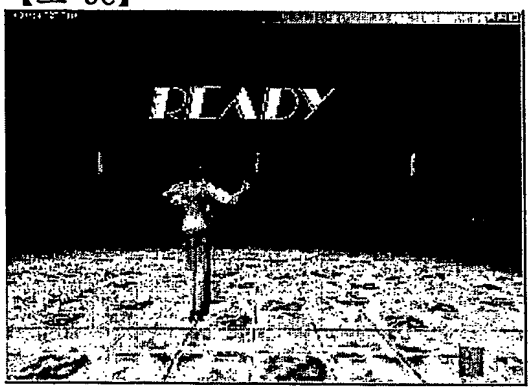
【도 9a】



【도 9b】



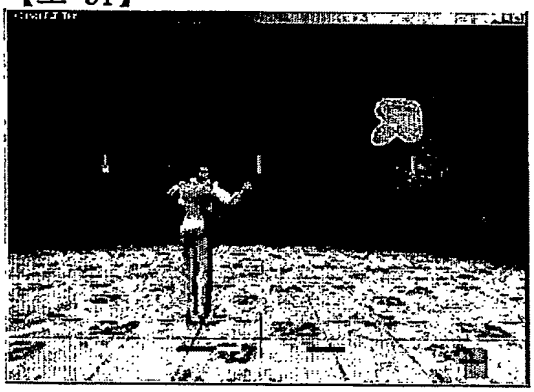
【도 9c】



【도 9d】

【도 9e】

【도 9f】



【도 9g】

【도 9h】